VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

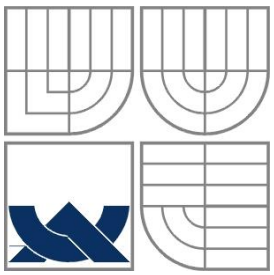
ZABEZPEČENÁ KOMUNIKACE PRO ZAŘÍZENÍ TYPU SMARTPHONE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

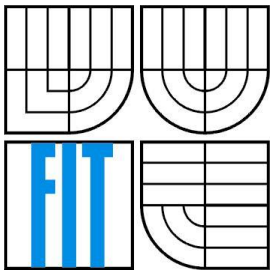
AUTOR PRÁCE
AUTHOR

TOMÁŠ TVRDOŇ

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ZABEZPEČENÁ KOMUNIKACE PRO ZAŘÍZENÍ TYPU SMARTPHONE

SECURE COMMUNICATION FOR SMARTPHONES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ TVRDOŇ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PAVOL KORČEK

BRNO 2013

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací aplikace pro zabezpečený přenos SMS zpráv. Dále je v texu zmapováno programovací rozhraní chytrých mobilních telefonů a metody šifrování. Hlavní důraz při návrhu aplikace byl kladen na uživatelsky přívětivé přidávání kontaktů, bezpečnou výměnu šifrovacích klíčů a snadné intuitivní ovládání.

Abstract

This bachelor thesis describes design and implementation of secure SMS application. Other parts of this thesis discuss Smartphone programming interface and cryptographic methods. Application design focuses on user friendly operations with contacts, secure encryption key exchange and intuitive user interfaces.

Klíčová slova

Android, chytrý telefon, SMS, šifrování, AES, RSA, uživatelské rozhraní

Keywords

Android, smartphone, SMS, encryption, AES, RSA, user interface

Citace

Tvrdoň Tomáš: Zabezpečená komunikace pro zařízení typu smartphone, bakalářská práce, Brno, FIT VUT v Brně, 2013

Zabezpečená komunikace pro zařízení typu smartphone

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Pavla Korčeka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Tomáš Tvrdoň
15. května 2013

Poděkování

Chtěl bych poděkovat vedoucímu mojí bakalářské práce Ing. Pavlovi Korčekovi za jeho čas a cenné připomínky při vývoji aplikace a vytváření této technické zprávy. Dále bych rád poděkoval kamarádům za pomoc s testováním výsledná aplikace.

Obsah

1 Úvod

Mobilní telefony se v dnešní době staly neodmyslitelnou součástí našeho života. Většina z nás je používá každodenně, především ke komunikaci. Primárním účelem těchto zařízení je uskutečňovat telefonní hovory, kdy již uživatel není vázán na pevný telefonní přístroj, ale může mobilní telefon využívat všude, kde je k dispozici pokrytí radiovým signálem.

Díky miniaturizaci a pokroku v technologiích jsou mobilní telefony skutečně kapesními zařízeními s mnoha dalšími funkcemi kromě telefonování. Moderní přístroje jsou již ekvivalentem počítačů, s výkonným hardwarem, vlastními operačními systémy a uživatelskými aplikacemi. Tato kategorie mobilních telefonů se označuje jako chytré telefony – smartphone. Mimo jiné umožnují zasílání textových a multimediálních zpráv, stálé připojení mobilního zařízení k síti internet, správu emailových účtů, GPS navigaci, přehrávání multimédií, poslech rádia atp.

Se stále většími možnostmi mobilních zařízení je pro řadu uživatelů důležité řešit otázku soukromí a zabezpečení informací. Cílem této práce je návrh zabezpečené komunikace pro zařízení typu smartphone. Pro mnoho uživatelů je nejčastějším využitím mobilního telefonu (vynechám-li jeho primární funkci jako telefonního přístroje) zasílání textových zpráv. Proto se v této práci budu dále věnovat otázce bezpečného přenosu SMS zpráv.

V úvodní části se zaměřím na programovací rozhraní mobilních zařízení, zejména platformy Google Android [1]. V druhé části potom na metody šifrování a popis šifrovacích algoritmů, které jsou v navrhované aplikaci použity. V další části jsou přiblíženy existující řešení pro zabezpečený přenos SMS zpráv pro platformu Google android.

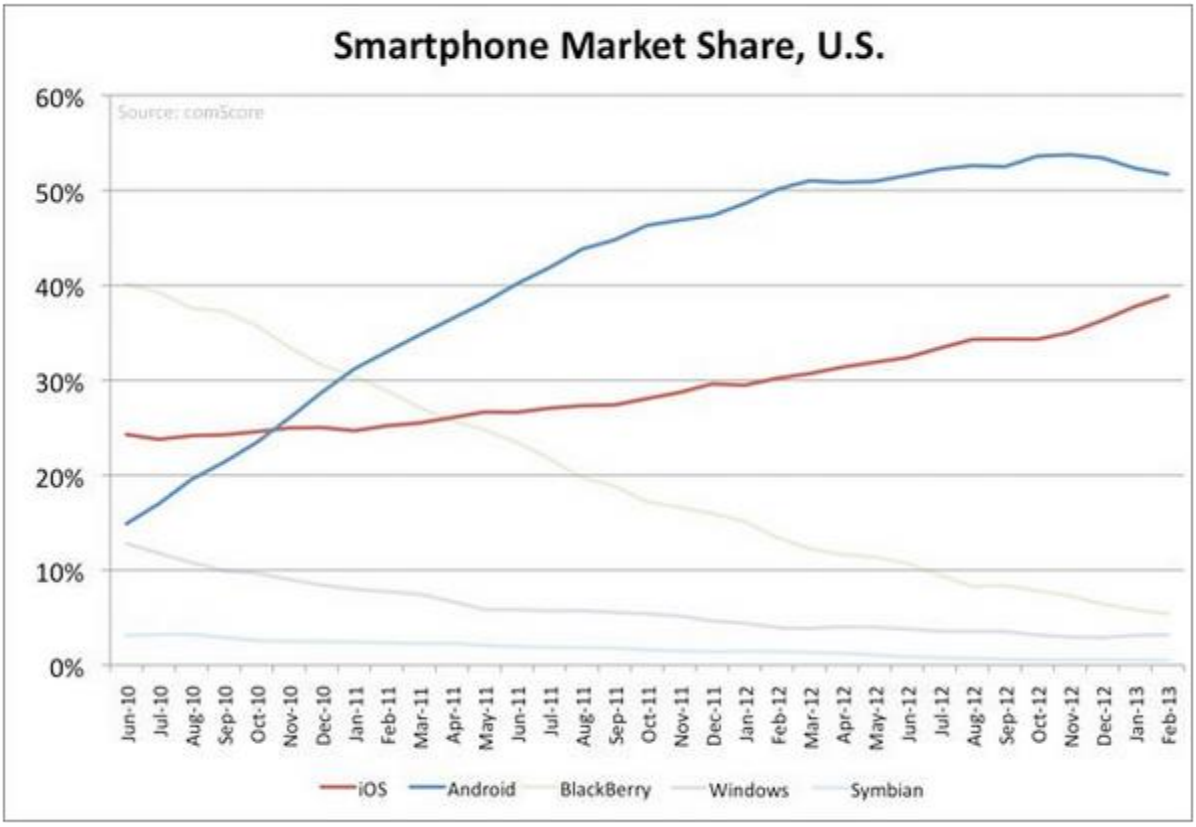
Hlavním tématem je návrh aplikace pro zabezpečení zasílání SMS zpráv, popis její implementace a testování pro platformu Google Android.

1.1 Podíl mobilních platforem na trhu

Existuje několik platforem a operačních systému pro chytré telefony. Při výběru zařízení je pro mnoho koncových uživatelů rozhodující dostupná technická podpora a množství a kvalita softwaru. Pro rozšířenější platformy je k dispozici více uživatelského softwaru, ze kterého je možno vybírat.

Pro vývojáře software pro mobilní platformy je rozhodující především její rozšíření a zastoupení na trhu, aby byl vyvíjený software dostupný co možná největšímu počtu potenciálních uživatelů.

Aktuálně nejrozšířenější je platforma Google Android, druhou nejrozšířenější je platforma Apple iOS, (viz obrázek 1.1, rozšíření mobilních platforem v USA). Na evropském trhu je situace obdobná. Pro český trh nejsou konkrétní data k dispozici, protože např. Český statistický úřad neposkytuje údaje pro využívání mobilních telefonů s dostatečnými detaily.



Obrázek 1.1: Podíl na trhu s chytrými telefony, převzato z [2].

2 Programování mobilních zařízení

Každý chytrý mobilní telefon přichází na trh s celou řadou předinstalovaných aplikací. Dále poskytuje uživateli možnost doinstalovat další aplikace od výrobců třetích stran. Aplikace jsou dostupné ze specializovaných online obchodů pro každou platformu. První takový obchod nabídla firma Apple pro zařízení iPhone – App Store[1]. Firma Nokia pro svá zařízení se systémem Symbian představila obchod OviStore[2]. Pro zařízení s operačním systémem Windows Phone je k dispozici Windows Marketplace[3].

Výrobci mobilních telefonů vydávají vývojářské balíčky (SDK – software development kit), které usnadňují tvorbu aplikací. Obsahují knihovny funkcí pro danou platformu, dokumentace rozhraní a jazyka, překladač a případně i emulátor pro usnadnění ladění aplikace.

V dalších kapitolách popíši architekturu a možnosti vývoje aplikací pro dvě nejrozšířenější platformy, iOS a Android (viz obrázek 1.1), a možnosti multiplatformního vývoje aplikací pro mobilní telefony.

2.1 Apple iOS

iOS je operační systém vytvořený společností Apple. Jedná se o sytém UNIXového typu vycházející z Mac OS X. Neobsahuje veškerou funkcionalitu OS X, ale přidává některé vlastnosti důležité pro mobilní zařízení, jako je podpora dotykového ovládání. Využívá se jak na mobilních telefonech iPhone, tak i na tabletech iPad a dalších mobilních zařízeních.

Firma Apple poskytuje iOS SDK [3] obsahující grafické vývojové prostředí Xcode, emulátor mobilního zařízení, optimalizační nástroje a nástroj pro tvorbu uživatelských rozhraní. Nástroje lze využívat pouze na zařízeních s operačním systémem Mac OS X. Vývoj aplikací na jiných platformách není možný. Výsledné aplikace nelze instalovat na telefony iPhone přímo, ale pouze přes již zmiňovaný obchod App Store po jejich schválení firmou Apple.

Programovacím jazykem pro tvorbu aplikací je Objective-C. Je to objektově orientovaný jazyk implementovaný jako rozšíření jazyka C o systém zasílání zpráv z jazyka Smalltalk. Je také možno programovat v jazyce C.

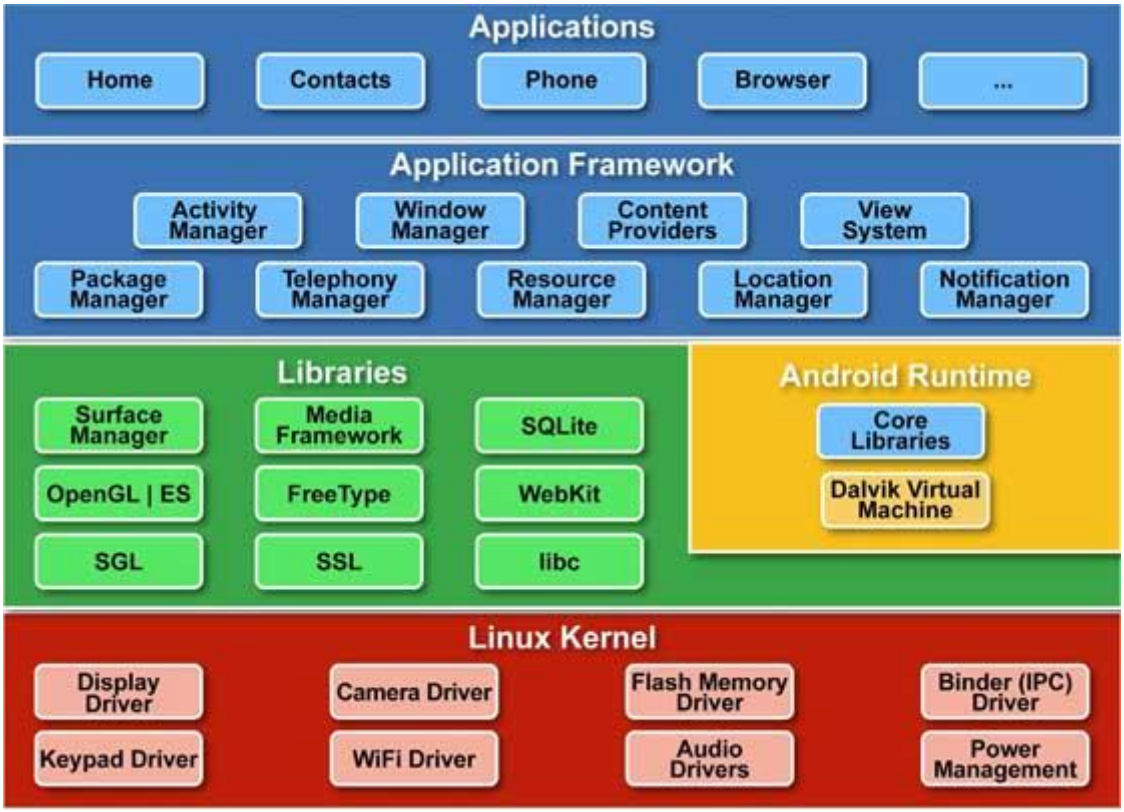
2.2 Google Android

Android je otevřená platforma vzniklá pro mobilní zařízení zahrnující operační systém UNIXového typu, uživatelské rozhraní a aplikace. Architektura je rozdělena do pěti základních vrstev. Popis architektury čerpá z [4], viz obrázek 2.1.

[1] Dostupné z URL: http://www.apple.com/osx/apps/app-store.html

[2] Dostupné z URL: http://store.ovi.com/

[3] Dostupné z URL: http://www.windowsphone.com/store



Obrázek 2.1: Architektura Android, převzato z [4].

- Nejnižší vrstvou je jádro operačního systému (*Linux Kernel*) postavené na OS Linux verze 2.6 bez podpory X Window systému a omezenou sadou GNU knihoven.
- Druhou vrstvou jsou knihovny (*Libraries*) v jazycích C/C++, které využívají ostatní komponenty systému. Jsou to například knihovny pro práci s SQLite databázemi, OpenSSL pro zabezpečení komunikace nebo OpenGL pro vykreslování 3D grafiky.
- Třetí vrstva *Android Runtime* obsahuje virtuální stroj Dalvik pro interpretaci aplikací v jazyce Java. Je to alternativa k Java virtual machine uzpůsobená a optimalizovaná k běhu na mobilních zařízeních. Další součástí této vrstvy jsou základní knihovny jazyka Java.
- Vrstva *Application Framework* poskytuje aplikacím přístup k systémovým službám, zdrojům a prvkům uživatelského rozhraní.
- Nejvyšší vrstvou jsou aplikace (*Applications*) využívané koncovými uživateli. Jedná se o předinstalované aplikace a dodatečně stažené aplikace třetích stran.

Pro vývojáře aplikací je k dispozici Android SDK [5]. Obsahuje vývojové prostředí Eclipse s předinstalovanými doplňky ADT (Android Developer Tools), nejnovější verzi platformy Android s kompletním systémovým obrazem pro emulátor, s knihovnami, dokumentací a ukázkovými programy. Lze také dodatečně stáhnout Android platformu kterékoliv starší verze systému. Vývojové nástroje jsou k dispozici pro všechny běžně používané operační systémy.

Programovacím jazykem pro platformu Android je Java. Jedná se o objektově orientovaný interpretovaný jazyk. Lze samozřejmě používat programy nebo jejich části napsané ve kterémkoliv jazyce spustitelném na operačním systému Linux (například C/C++), které je nutné obalit spouštěčem v jazyce Java.

Ladění aplikací umožnuje debugger, který je součástí Android SDK. Aplikace lze ladit jak na emulátoru mobilního telefonu, tak na jakémkoliv přístroji s operačním systémem Android

připojeném přes rozhraní USB. Je možné použít více zařízení zároveň (např. více emulátorů) pro testování a ladění komunikace.

Java je interpretovaný jazyk, proto neprobíhá překlad aplikace, ale kompilace do Java byte kódu pro interpret *Dalvik*. Výsledkem kompilace je *apk* soubor určený k instalaci.

Výsledné aplikace lze instalovat buď prostřednictvím obchod Google Play [6] nebo přímo zkopírováním do telefonu a spuštěním instalace. Instalaci z jiných zdrojů než obchod Play je nutno povolit v nastavení telefonu.

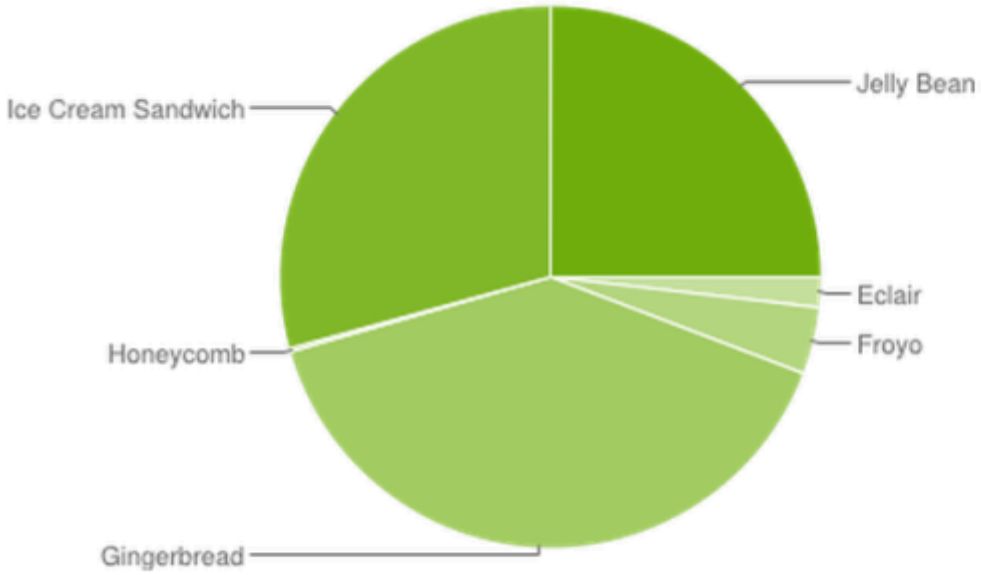
Aplikace pro platformu Android se mohou skládat ze základních čtyř typů komponent [7].

1. Activity – odpovídá jedné obrazovce aplikace, obsahuje grafické uživatelské rozhraní pro interakci s uživatelem. Aplikace obvykle obsahují více aktivit, mezi kterými uživatel přepíná. Aktivity si mezi sebou mohou předávat různé informace. Každá aktivita obsahuje funkční kód a popis rozložení grafického uživatelského rozhraní v XML formátu. Životní cyklus aktivit a jejich přepínání má na starosti Activity Manager, který je součástí vrstvy Application Framework.
2. Service – představuje proces běžící na pozadí. Využívá se k provádění časově náročných operací a přístupu ke vzdáleným zdrojům s předem neznámou dobou odezvy.
3. Content Provider – rozhraní pro sdílení dat mezi aplikacemi nebo mezi jednotlivými aktivitami jedné aplikace. Umožnuje přístup k datům pomocí metod podobným databázovým metodám insert, update, delete, query.
4. Broadcast receiver – komponenta určená k příjmu oznámení a jejich zpracování na pozadí. Aplikace mohou využívat systémová oznámení (např. příchozí zpráva, příchozí hovor, nízký stav baterie) nebo vytvářet a přijímat vlastní oznámení.

Jednotlivé komponenty jsou aktivovány zasláním asynchronní zprávy, nazývané *Intent*. Zprávy může aplikace zasílat jak svým ostatním komponentám, tak i jiné aplikaci. Tyto zprávy mohou obsahovat informace potřebné pro běh komponenty (například zpráva s příznakem „send“ a řetězcem pro vyhledávání zaslaná aplikaci pro správu kontaktů znamená, že tato aplikace se pokusí najít kontakt podle vyhledávacího řetězce a odkaz na něj zaslat pomocí Intentu zpět). Pomocí Intentů se spouštějí aktivity a služby; intenty obsahující oznámení spouští příslušné *Broadcast receivers*.

Každá Android aplikace musí obsahovat AndroidManifest soubor ve formátu XML, ve kterém je definována struktura aplikace a jednotlivé použité komponenty, název aplikace, údaje o verzi aplikace, vyžadovaná verze platformy Android a seznam speciálních knihoven API, které aplikace využívá (například knihovna Google maps). Dále zde jsou definována práva potřebná pro běh aplikace. Tato informace je potřebná zejména pro koncového uživatele aplikace, který při instalaci vidí a potvrzuje, jaká práva daná aplikace vyžaduje. Jsou to například: připojení k síti internet, příjem a odesílaní zpráv, přístup k seznamu kontaktů, ovládání displeje, vyzvánění, vibrace atd.

Každá novější verze platformy android přináší vždy řadu nových možností pro vývojáře aplikací. Bohužel ne všechny novinky lze použít i pro starší verze systému. Je třeba si zvolit, pro kterou minimální verzi platformy android bude aplikace vyvíjena a podle toho přizpůsobit použité technologie a funkce API Android. Na obrázku 2.2 je graf rozšíření jednotlivých verzí platformy Android, v tabulce 2.1 potom i popis jednotlivých verzí platformy a příslušné číslo verze Android API. Údaje jsou platné k dubnu 2013.



Obrázek 2.2: Rozšíření verzí platformy Android, převzato z [8].

Tabulka 2.1: Rozšíření verzí platformy Android, převzato z [8].

Verze	Označení	API	Rozšíření
1.6	Donut	4	0.1 %
2.1	Eclair	7	1.7 %
2.2	Froyo	8	4.0 %
2.3 – 2.3.2	Gingerbread	9	0.1 %
2.3.3 – 2.3.7		10	39.7 %
3.2	Honeycomb	13	0.2 %
4.0	Ice Cream Sandwich	15	29.3 %
4.1	Jelly Bean	16	23.0 %
4.2		17	2.0 %

2.3 Nástroje pro multiplatformní vývoj aplikací

Při vývoji aplikací pro mobilní zařízení je často třeba cílit vývoj na více mobilních platforem. Toho je možné dosáhnout několika způsoby.

První možností je vývoj několika nezávislých aplikací se stejnou funkcionalitou, zvlášť pro každou platformu. Tento přístup je použitelný u velmi malých projektů, u kterých se nepočítá s následným rozvojem. Při větších projektech je problémem velká časová náročnost tohoto přístupu a špatné možnosti udržování několika nezávislých projektů.

Jinou možností je využití nástrojů pro multiplatformní vývoj. Tyto nástroje umožnují vyvíjet a nasazovat aplikace pro více platforem současně. Snižují náklady na nasazení aplikace na novou platformu a umožnují vysokou znuvupoužitelnost zdrojových kódů.

V následujících kapitolách zmíním nejpoužívanější techniky multiplatformního vývoje aplikací pro mobilní zařízení.

2.3.1 Webové technologie

Pro vývoj aplikací lze využít známé webové technologie, jako je HTML (dnes nejčastěji ve verzi 5), CSS a JavaScript. Aplikace potom může být čistě webová, optimalizovaná pro zobrazení na mobilních zařízeních a umístěná na internetovém serveru. Zařízení poté aplikaci zobrazuje ve webovém prohlížeči jako klasickou stránku.

Nevýhodami tohoto přístupu jsou nutnost uživatele využívat při práci s aplikací připojení k síti internet, latence aplikace závislá na rychlosti připojení a především nutnost optimalizovat aplikaci pro velké množství používaných internetových prohlížečů s různými zobrazovacími jádry.

Výše zmíněné problémy se snaží řešit hybridní aplikace. Jádro aplikace, stejně jako při předchozím přístupu, je napsáno pomocí webových technologií. Toto jádro, společné pro různé platformy je poté obaleno nativním pouzdrem (spouštěčem) pro každou platformu. Používání takovýchto aplikací se nijak neliší od aplikací nativních, instalují se stejně a nevyžadují pro běh připojení k síti internet.

Pro vývoj hybridních aplikací jsou k dispozici knihovny, poskytující nativní API podporovaných platforem pro jádro aplikace v jazyce JavaScript. Mezi nejznámější patří PhoneGap[4] nebo jQuery mobile[5] [9].

2.3.2 C# a .NET Framework

Dalším dostupným prostředkem je nástroj Xamarin [10]. Tento framework poskytuje nativní API funkce platforem Android, iOS a Windows Phone pomocí knihoven pro Microsoft .NET. Vývojářům je k dispozici Xamarin Studio s integrovaným designérem GUI, případně doplněk pro Microsoft Visual Studio. Vývoj aplikací poté probíhá v dnes velmi rozšířeném jazyce C#.

Výsledné aplikace vyžadují na cílovém zařízení interpret pro Microsoft CLI (Common Language Infrastructure). Na platformě Windows Phone je to nativní .NET framework. Implementacemi pro Android a iOS jsou Mono a Xamarin.iOS[6].

[4] Dostupné z URL: http://phonegap.com/

[5] Dostupné z URL: http://jquerymobile.com/

[6] Dostupné z URL: http://www.mono-project.com/Main_Page

3 Kryptografie

Tato kapitola čerpá z [11, 12]. Kryptografie je matematický vědní obor, který se zabývá metodami utajování smyslu zpráv převodem do podoby, která je čitelná jen se speciální znalostí. Název pochází z řečtiny: *kryptós* je skrytý a *gráphein* znamená psát. Obor se zabývá šifrovacími a kódovacími algoritmy.

- Šifrovací algoritmus – algoritmus, který se snaží utajit data jejich šifrováním. K šifrování se využívá tajemství, které většinou nazýváme šifrovací klíč.
- Kódovací algoritmus – algoritmus snažící se chránit data před neoprávněným přečtením. O utajení se stará samotný algoritmus, bez použití tajemství.

Cílem šifrování je převést data v otevřené podobě (otevřený text) do utajené podoby (šifrovaný text). Převod opačným směrem se nazývá dešifrování.

Šifrovací algoritmy dělíme podle několika hledisek. Jedním z nich je klasifikace podle množství dat, která jsou v daném okamžiku šifrována. Proudové šifry šifrují otevřený text po znacích. Je-li otevřený text šifrován po větších blocích, jedná se o blokové šifry.

Druhou klasifikací je dělení podle typu použitého šifrovacího klíče. Pokud se pro šifrování i dešifrování používá stejný klíč, jedná se o symetrickou kryptografii. Asymetrická kryptografie používá pro šifrování jiný klíč než pro následné dešifrování.

3.1 Symetrická kryptografie

Algoritmy symetrické kryptografie, někdy též nazývané konvenční, pracují s jedním klíčem, společným pro obě komunikující strany. Tento klíč slouží jak k zašifrování otevřeného textu, tak k jeho dešifrování. Výhodou symetrické kryptografie je její vysoká rychlost (ve srovnání s asymetrickými metodami) a nízká výpočetní náročnost. Nevýhodou jsou vyšší nároky na správu klíčů. Každé dva komunikující subjekty sdílejí jeden klíč. S každým dalším komunikujícím subjektem roste počet klíčů v systému exponenciálně. Zároveň je třeba zajistit bezpečné sdílení klíčů mezi komunikujícími stranami.

Symetrické šifrovací algoritmy využívají dvě základní techniky. Substituci a transpozici.

3.1.1 Substituční šifry

Substituční šifra nahrazuje znaky otevřeného textu znaky šifrovaného textu podle předepsaného klíče. Přiřazení znaků šifrovaného textu znakům otevřeného textu je závislé na hodnotě šifrovacího klíče a řídí se substituční tabulkou.

Monoalfabetické šifry jsou nejjednodušší variantou substitučních šifer. Používají statickou tabulku vygenerovanou podle šifrovacího klíče. Nedostatkem této metody je možnost využití frekvenční charakteristiky pro prolomení šifry. Pokud je šifrovaný text dostatečně dlouhý, dokáže tato analýza odhalit, který znak byl za který nahrazen. Znaky v šifrovaném textu budou zastoupeny se stejnou frekvencí jako v textu otevřeném. Nejznámější šifrou z této skupiny je Césarova šifra. Spočívá v posunu abecedy o tři znaky.

Nevýhodu monoalfabetických šifer odstraňují homofonní substituční šifry. Každému znaku otevřeného textu může být přiřazeno více různých znaků šifrovaného textu.

Nejpokročilejšími substitučními šiframi jsou polyalfabetické šifry. Jedná se o skupinu monoalfabetických šifer, které jsou postupně aplikovány na jednotlivé znaky otevřeného textu. Příkladem je takzvaná Vignerova šifra. Základem je čtvercová substituční tabulka, kde v prvním řádku jsou všechny znaky otevřeného textu, v každém dalším řádku potom stejné znaky posunuté vždy o jednu pozici. Šifrovací klíč určuje, které řádky se při šifrování použijí.

Za druhé světové války byl německou armádou využíván šifrovací stroj Enigma. Podstatou byla skupina rotorů, kde každý z nich prováděl jednu substituci. Znak otevřeného textu procházel postupně přes několik rotorů točících se různou rychlostí. I pře tuto složitost se podařilo šifru prolomit.

Existuje jeden typ šifry, patřící mezi substituční, který je považován za nerozluštitelný. Jedná se o takzvanou jednorázovou tabulku. K šifrování se používá náhodný klíč o stejné délce, jako je samotný otevřený text. Příkladem je Vernamova šifra, která provádí bitovou operaci nonekvivalence nad každým znakem otevřeného textu s příslušným znakem šifrovacího klíče. Dešifrování probíhá identickým postupem. Problémem je reálná nemožnost vytvořit naprosto náhodný klíč (pseudonáhodné generátory jsou nedostačující), jeho délka, přenos klíče mezi komunikujícími stranami bezpečným způsobem a nutnost vytvořit nový klíč pro každou zprávu.

3.1.2 Transpoziční šifry

Transpozice nezaměňuje znaky otevřeného textu za jiné, ale mění jejich pozici v šifrované zprávě. Pro tento postup se používá matematický termín permutace.

Nejjednodušší transpozicí je rozepsání otevřeného textu do sloupců s danou délkou a jeho následné přepsání po řádcích. Při takto jednoduché permutaci stačí k rozluštění najít periodu, se kterou byl otevřený text přepsán.

Lze použít jakýkoliv reverzibilní permutační algoritmus. Výhodou tohoto typu šifry je rozbití větších struktur z otevřeného textu (bigramy – dvojice znaků, trigramy – trojice znaků).

3.1.3 Algoritmus AES

Šifra AES (Advanced Encryptin Standard) od autorů Joada Daemena a Vincenta Rijmena, také známá pod názvem Rijndael je šifrovací algoritmus, využívající jak substituční, tak transpoziční principy. Algoritmus byl přihlášený do veřejné soutěže NIST o americký federální šifrovací algoritmus z roku 1997. Národní bezpečnostní agenturou NSA byl uznan ke kódování nejtajnějších dokumentů a od roku 2002 začal být používán jako federální standard USA.

Jedná se o symetrickou blokovou šifru s délkou bloku 128 bitů (16 bytů) a velikostí klíče 128, 192 nebo 256 bitů. Algoritmus pracuje s maticí 4x4 bytů označovanou jako stav a skládá se ze čtyř částí.

1. Expanze klíče – z šifrovacího klíče jsou odvozeny podklíče.
2. Inicializační část – každý byte stavu je zkombinován s podklíčem pomocí nonekvivalence nad všemi bity.
3. Iterace
   - Záměna bitů – nelineární substituční krok využívající vyhledávací tabulku.
   - Transpozice řádků stavu.
   - Transpozice sloupců stavu podle speciální matice.
   - Kombinace stavu s podklíčem.
4. Závěrečná část – stejná jako iterace, pouze nedochází k transpozici sloupců stavu.

Podle délky šifrovacího klíče je s každým stavem provedeno 10 (128bit), 12 (192bit) nebo 14 (256bit) iterací.

Prolomení algoritmu hrubou silou (vyzkoušení všech možných kombinací klíče) je s dnešními výpočetními možnostmi nemožné, útok na šifru s 256 bit klíčem by vyžadoval 2^{256} operací.

3.2 Asymetrická kryptografie

Asymetrická kryptografie používá pro šifrování a dešifrování odlišné šifrovací klíče – klíčový pár. Odesílatel použije k zašifrování zprávy jednu část klíče, často veřejně známou, označovanou jako veřejný klíč. Tímto klíčem již není možno zprávu dešifrovat. K dešifrování zprávy se použije druhá část klíče, která je známa pouze příjemci zprávy a označuje se jako soukromý klíč. Obě strany komunikace nemusí sdílet žádné tajemství, čímž je eliminována potřeba bezpečné výměny klíčů. Při větším počtu komunikujících subjektů je potřeba pouze dvounásobný počet klíčů, na rozdíl od exponenciálního počtu klíčů u symetrických metod.

Šifrovací a dešifrovací klíče musí být matematicky svázány, podmínkou je nemožnost odvodit ze znalosti šifrovacího klíče dešifrovací klíč. Tyto metody kryptografie jsou založeny na jednocestných funkcích (operace, které lze snadno provést v jednom směru). Ze vstupu funkce je snadné spočítat výstup, z výstupu je však obtížné nalézt vstup.

Asymetrickou kryptografii lze použít i pro digitální podpis zpráv. Princip spočívá ve vytvoření otisku podepisované zprávy (hash), který je zašifrován soukromým klíčem odesílatele a přiložen k původní zprávě. Příjemce zprávy otisk dešifruje veřejným klíčem odesílatele, vytvoří svůj vlastní otisk zprávy a oba otisky porovná. Pokud se shodují, je zřejmé, že zpráva nebyla při přenosu změněna a je ověřena identita odesílatele.

3.2.1 Algoritmus RSA

RSA je algoritmus pro asymetrickou kryptografii publikovaný roku 1978, pojmenovaný po jeho tvůrcích (Ronald Rivest, Adi Shamir a Leonard Adleman), založený na Eulerově větě a faktorizaci prvočísel. Je použitelný jak pro šifrování, tak pro digitální podpis.

Před samotným šifrováním je potřeba vygenerovat soukromý a veřejný klíč [13].

Zvolíme si dvě náhodná velká prvočísla p_a a q_a a jejich součin označíme jako n_a. Následně si zvolíme číslo e_a nesoudělné s $\varphi(n_a)$, kde

$$\varphi(n_a) = \varphi(p_a \cdot q_a) = (p_a - 1) \cdot (q_a - 1), \tag{3.1}$$

Kde $\varphi(n)$ je hodnota Eulerovy funkce pro n a zároveň platí

$$e_a < \varphi(n_a). \tag{3.2}$$

Nakonec vypočítáme číslo d_a, které je multiplikativní inverzí čísla e_a. Potom dvojice (n_a, e_a) je veřejným klíčem. Část klíče n_a se nazývá modul, e_a šifrovací či veřejný exponent. Dvojice (n_a, d_a) je soukromým klíčem, část d_a se nazývá dešifrovací či soukromý exponent.

K zašifrování otevřené zprávy z použijeme veřejný klíč podle vztahu

$$x = z^{e_a} \bmod(n_a), \tag{3.3}$$

kde *x* je šifrovaná zpráva. Dešifrování provedeme analogicky podle vztahu

$$z = x^{d_a} \bmod(n_a). \tag{3.4}$$

Reálné algoritmy pro výpočet RSA klíčů pracují na principu generování velkých náhodných čísel a jejich následným testováním na prvočíselnost. Generování klíčů je matematicky a výpočetně náročné, stejně tak i samotné šifrování a dešifrování.

Pro bezpečnost tohoto algoritmu je důležité zvolit dostatečně velká čísla p_a a q_a tak, aby případná faktorizace jejich součinu byla velmi náročná.

V praxi se často používá kombinace symetrických a asymetrických algoritmů. Asymetrická šifra je použita pro bezpečnou výměnu klíče pro symetrickou šifru. Jedna z komunikujících stran zašle druhé straně svůj veřejný šifrovací klíč. Druhá strana vygeneruje šifrovací klíč pro symetrický šifrovací algoritmus, zašifruje ho pomocí obdrženého veřejného klíče a odešle zpět první straně. K samotné bezpečné komunikaci se poté používá symetrická šifra, která je rychlejší a vhodnější k šifrování většího množství dat. Na podobném principu pracuje například protokol SSL.

4 Existující aplikace pro zabezpečený přenos SMS zpráv

Aplikace pro platformu Google Android jsou uživatelům k dispozici prostřednictvím internetového obchodu Google Play, přístupného jak z webových stránek, tak především pomocí specializované aplikace na mobilních telefonech s operačním systémem Google Android.

V tomto obchodě je k dispozici několik aplikací pro zabezpečený přenos SMS zpráv. Tyto aplikace se liší uživatelskými rozhraními i nabízenými funkcemi. Funkčně se liší zejména metodami šifrování SMS zpráv a systémem přenosu šifrovacího klíče.

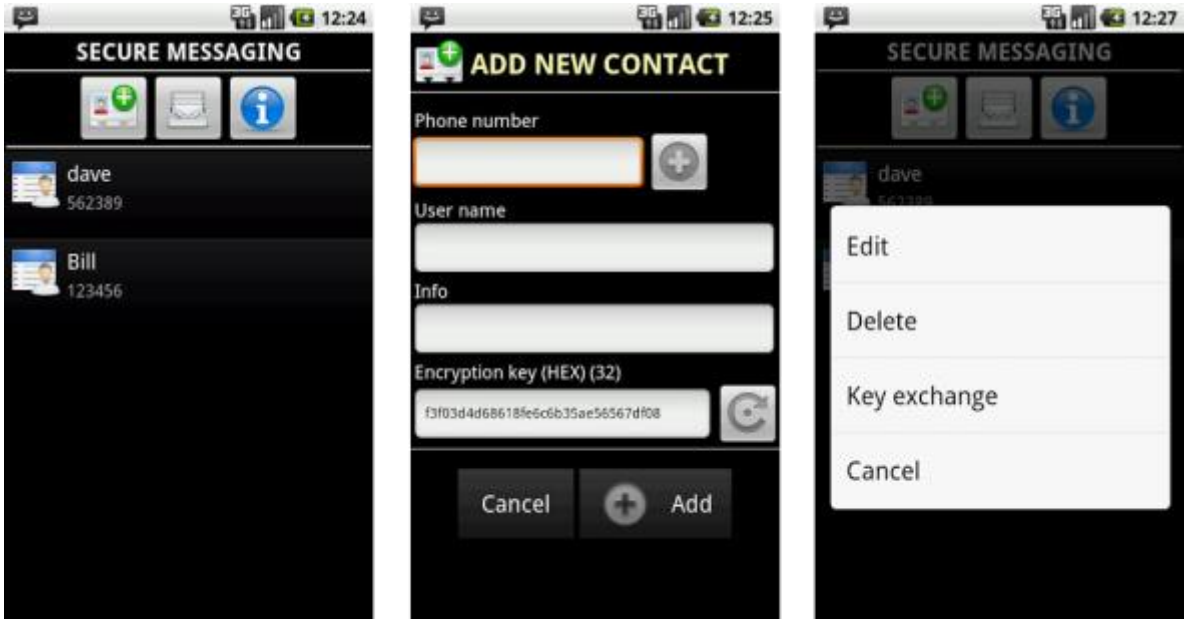
Z nabízených aplikací vyberu dvě nejzajímavější, především nabízenými funkcemi. Ostatní dostupné aplikace jsou především co do funkcionality velmi podobné vybraným dvěma.

4.1 Secure SMS – dDave

Tato aplikace od autora dDave slouží k odesílání a příjmu zašifrovaných SMS zpráv. Umožňuje přidání kontaktů ze stávajícího telefonního seznamu v mobilním telefonu. Při vytváření nového kontaktu je vygenerován šifrovací klíč. Z nabídky v seznamu kontaktů je poté možno zaslat klíč danému kontaktu pomocí SMS zprávy. Tato je bohužel velmi velká (cca 4 SMS zprávy v rozsahu 160 znaků) a není aplikací na straně příjemce nijak zpracována. Uživatel musí obdržený klíč ze zprávy zkopírovat a vložit ke kontaktu.

Po úspěšném přidání kontaktu oběma komunikujícími stranami je možno zasílat šifrované SMS zprávy. Šifrování probíhá s použitím algoritmu AES. Zašifrovaný text je přenášen hexadecimálně, zprávy jsou přibližně 2-3 krát delší než původní otevřený text. Vzhled aplikace viz obrázek 4.1 [14].

Výhodou této aplikace je možnost zaslání šifrovacího klíče SMS zprávou, nevýhodou je velikost tohoto klíče a velikost zasílaných šifrovaných zpráv.



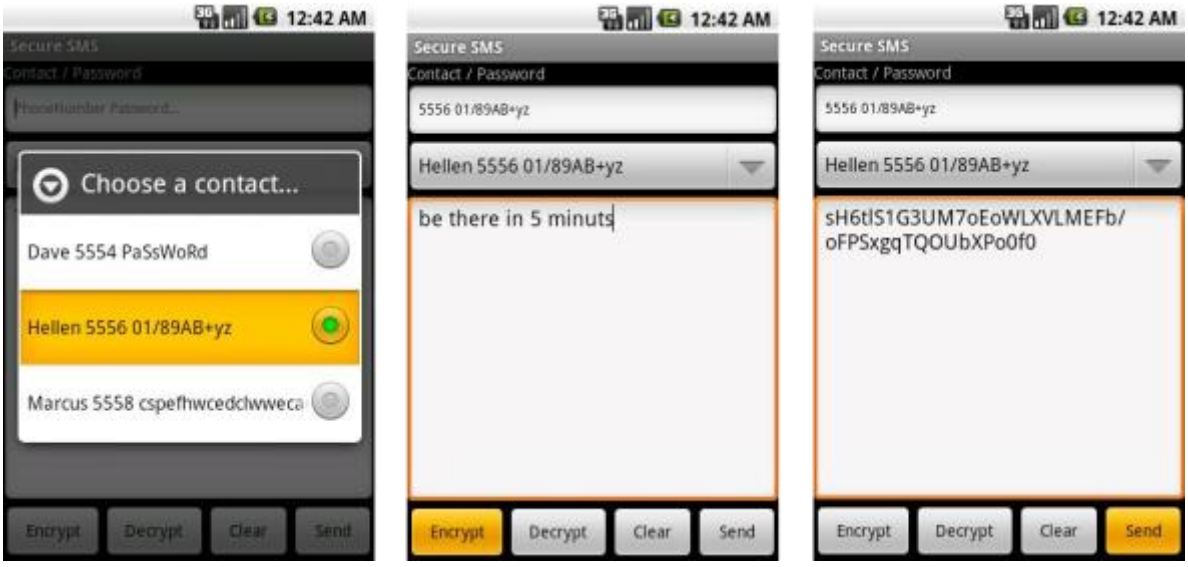
Obrázek 4.1: aplikace Secure SMS (dDave), převzato z [14].

4.2 Secure SMS – LegreSoft Inc.

Druhou vybranou aplikací pro zabezpečení SMS zpráv je Secure SMS od společnosti LegreSoft Inc. Podobně jako v předchozí zmiňované i zde si uživatel vytváří vlastní seznam kontaktů. Údaje o kontaktu musí mít přesně stanovenou podobu (jméno, telefonní číslo a šifrovací klíč, oddělené právě jednou mezerou). Aplikace šifruje SMS zprávy dvěma algoritmy zároveň, nejprve AES s 256 bitovým klíčem a poté algoritmem Serpenet, také s 256 bitovým klíčem. Šifrovací klíč zadaný u kontaktu je rozdělen na dvě 256 bitové části (pro každou z metod jedna část). Pokud je kratší, je doplněn nulami, pokud delší, je zkrácen.

Po napsání zprávy ji uživatel pomocí tlačítka zašifruje a odešle. Výsledná zašifrovaná zpráva je přibližně dvakrát delší než zpráva původní. Vzhled aplikace viz obrázek 4.2 [15].

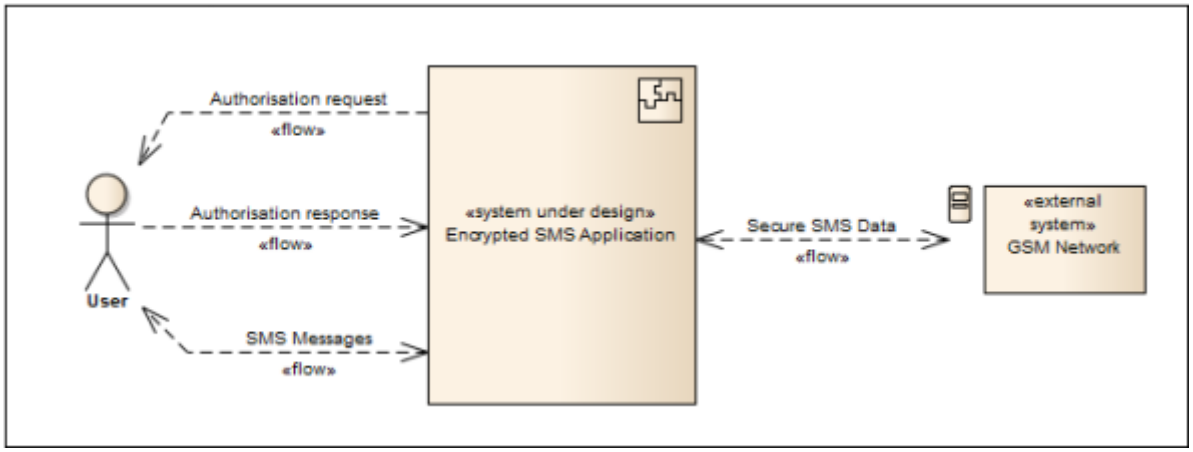
Výhodou této aplikace je velmi jednoduché používání. Nevýhodami jsou nutnost přesně dodržet formát jména kontaktu a pro dosažení optimální délky klíče nutnost zadat velmi dlouhý (512 bitů) klíč. Také šifrování dvěma velmi výkonnými symetrickými šifrovacími algoritmy je zbytečné, při takto zvolené délce šifrovacího klíče by zpráva byla těžko dešifrovatelná i při použití pouze jedné z metod.



Obrázek 4.2: aplikace Secure SMS (LegreSoft Inc.), převzato z [15].

5 Návrh aplikace

Cílem práce je vytvořit aplikaci pro zabezpečený přenos SMS zpráv. Aplikace zmíněné v předchozí kapitole nevyhovují systémem zadávání či výměny šifrovacích klíčů a zejména velikostí výsledných zašifrovaných SMS zpráv. Navrhovaná aplikace bude nainstalovaná na mobilním zařízení s platformou Android, ovládaná přímo uživatelem a bude komunikovat prostřednictvím GSM sítě, viz obrázek 5.1.



Obrázek 5.1: Koncept aplikace.

Aplikace umožňuje uživateli spravovat seznam kontaktů, bezpečně si s ostatními uživateli aplikace na jiných zařízeních vyměňovat šifrovací klíče a poté umožnit zasílání šifrovaných SMS zpráv.

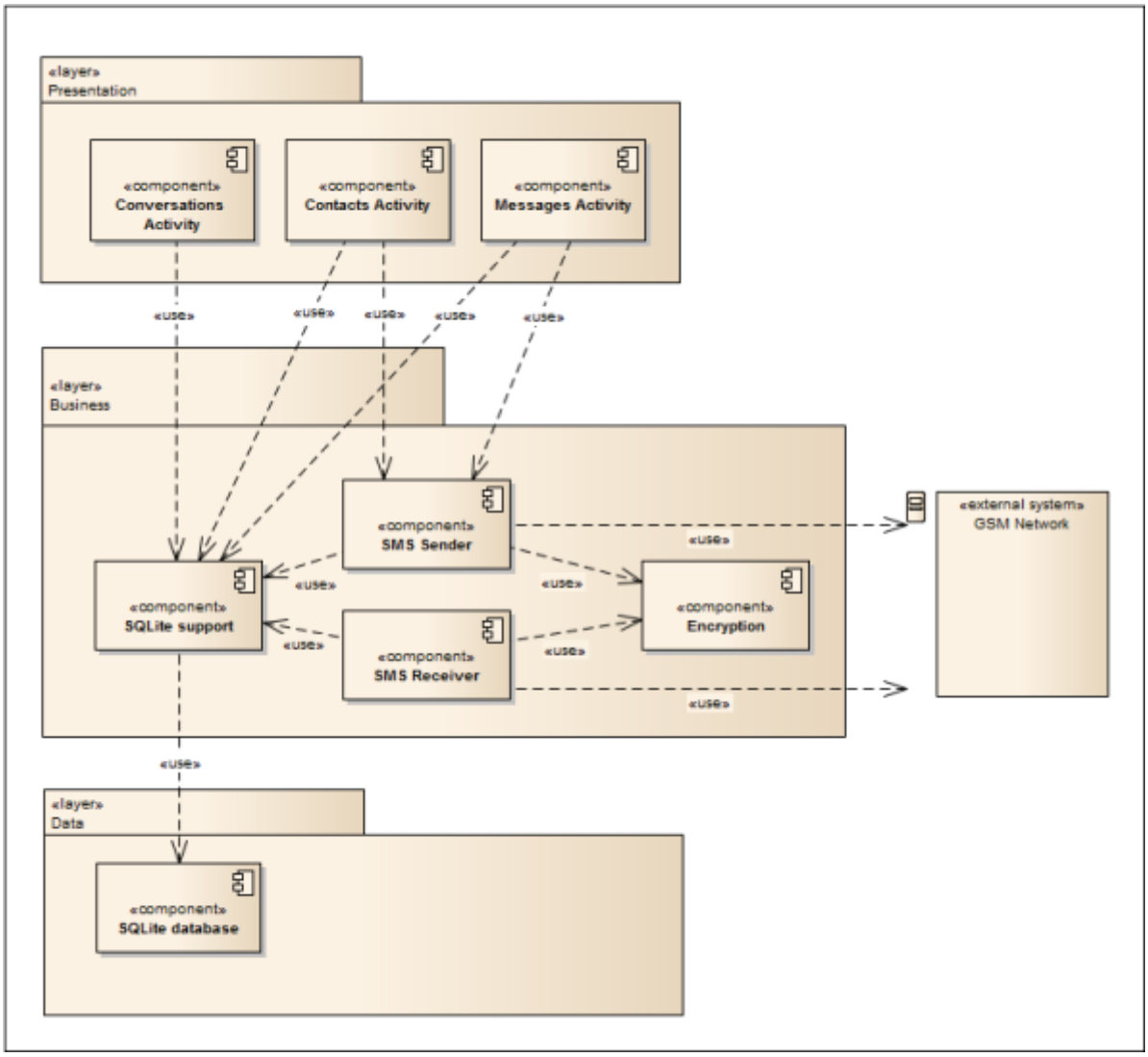
SMS zprávy budou šifrovány pomocí algoritmu AES. Tento algoritmus je dostatečně rychlý a poskytuje vysokou úroveň zabezpečení. Pro účely tohoto algoritmu bude použit nejdelší možný šifrovací klíč o délce 256 bitů. Výměna klíčů mezi uživateli bude řešena následujícím způsobem:

Při instalaci aplikace bude vygenerován klíčový pár pro šifrovací algoritmus RSA s délkou klíče 1024 bitů (delší klíč by zajišťoval vyšší úroveň zabezpečení, ale nebylo by možno jeho veřejnou část odeslat pomocí jedné SMS zprávy). Soukromý i veřejný klíč budou uloženy v databázi aplikace. Při přidání nového kontaktu aplikace odešle žádost o autorizaci obsahující veřejný RSA klíč odesílatele pomocí jedné SMS zprávy. Přijímací strana potvrdí autorizaci, vygeneruje 256 bitový AES šifrovací klíč, zašifruje ho veřejným RSA klíčem odesílatele a odešle zpět, znovu pomocí jedné SMS zprávy. Současně kontakt uloží spolu s vygenerovaným AES šifrovacím klíčem do databáze. Příjemce obdrží zašifrovaný AES klíč, pomocí svého soukromého RSA klíče ho rozšifruje a uloží ke kontaktu. V tuto chvíli mají obě komunikující strany k dispozici šifrovací klíč pro šifrování SMS zpráv. Postup výměny klíče je dále zmíněn v kapitole 5.2.

Způsob výměny klíčů je vhodný pro mobilní zařízení (nevyžaduje přímý kontakt komunikujících stran, pouze zaslání jedné SMS zprávy oběma stranami) a klíč je mezi oběma subjekty vyměněn bezpečným způsobem. Není možno zachytit šifrovací AES klíč v otevřené podobě třetí osobou.

5.1 Architektura aplikace

Architektura aplikace se skládá ze tří základních vrstev. Jedná se o prezentační vrstvu, aplikační vrstvu a datovou vrstvu, viz obrázek 5.2.



Obrázek 5.2: Architektura aplikace.

Prezentační vrstva zobrazuje uživateli informace pomocí grafického uživatelského rozhraní. V navrhované aplikaci se jedná o zobrazení seznamu kontaktů a jejich správu, seznamu konverzací s jednotlivými kontakty a seznam samotných přijatých a odeslaných zpráv. Téměř všechny aplikace pro mobilní telefony umožňující komunikaci pomocí SMS zpráv nezobrazují uživateli všechny přijaté či odeslané zprávy dohromady, ale řadí je do konverzací podle kontaktu (telefonního čísla). Navrhovaná aplikace bude zprávy zobrazovat stejným způsobem, aby usnadnila uživatelům orientaci ve zprávách. Další součástí prezentační vrstvy bude nastavení aplikace a nápověda.

Aplikační vrstva aplikace obsahuje jádro aplikace, veškerou logiku a zpracování dat.

Komponenta SQLite support má na starosti zpracování dat, obsahuje metody pro ukládání dat do databáze a jejich načítání pro účely prezentační vrstvy. Při instalaci aplikace vytvoří tato

komponenta novou databázi, při přechodech na vyšší verzi má na starosti aktualizaci databázové struktury. Dále je využívána komponentami pro odesílání a příjem SMS zpráv.

SMS Sender je část aplikační vrstvy zajišťující odesílání SMS zpráv. API platformy android nabízí možnost odesílání textových SMS zpráv, dlouhých textových zpráv a datových SMS zpráv. Datové zprávy mohou obsahovat až 133 bytů libovolných dat, není zde zohledněno kódování textu jako u standardních SMS zpráv. Na začátek textových SMS zpráv, pokud obsahují speciální znaky nebo diakritiku, je nutno přidat hlavičku s údaji o kódování, která omezí množství uživatelského textu na 70 znaků, standardní délka je maximálně 160 znaků. SMS sender využívá datových SMS zpráv pro zasílání požadavků na autorizaci (RSA veřejný klíč), potvrzení autorizace (zašifrovaný AES klíč) a samotných šifrovaných SMS zpráv pomocí GSM sítě.

SMS Receiver přijímá datové SMS zprávy z GSM sítě a ukládá je do databáze a notifikuje uživatele o přijetí zpráv.

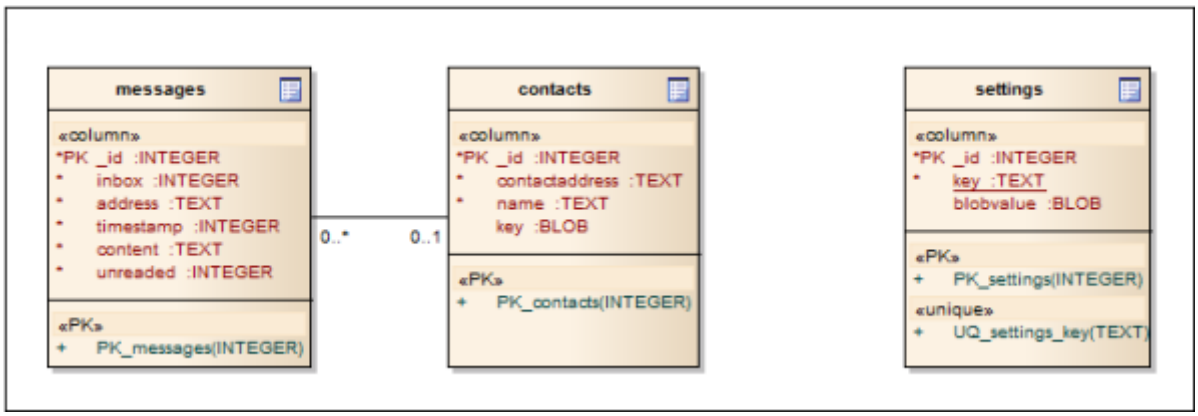
Typ odesílané a přijaté zprávy je rozlišován podle hlavičky, podrobněji zmíněno v kapitole 6.

Komponenta Encryption umožňuje generování RSA a AES šifrovacích klíčů, jejich transformaci z objektové podoby do pole bytů a zpět (pro účely uložení v databázi a přenosu pomocí SMS zpráv). Hlavní funkcí je šifrování a dešifrování pomocí RSA a AES algoritmu.

Datovou vrstvu tvoří SQLite databáze [16]. Pro práci s SQLite databázemi poskytuje API Android specializovanou knihovnu a jejich využití je běžné v mnoha aplikacích pro Android. Databáze jsou uloženy v paměti mobilního zařízení s právy pro přístup pouze pro danou aplikaci, data jsou tudíž bezpečně uložena.

5.2 Databázový model

Na obrázku 5.3 je znázorněn ER diagram navrhované databáze pro ukládání dat aplikace.



Obrázek 5.3: Model databáze.

Data budou uložena ve třech tabulkách v SQLite databázi.

Tabulka *settings* obsahuje nastavení systému, zejména při instalaci vygenerovaný klíčový pár pro RSA šifrovací algoritmus sloupec *key* obsahuje klíč a je unikátní. Ve sloupci *blobvalue* je potom ke klíči přiřazena libovolná hodnota daného nastavení.

Kontakty jsou ukládány do tabulky *contacts*. Sloupec *contactaddress* obsahuje telefonní číslo kontaktu, *name* jméno a ve sloupci *key* je uložen vygenerovaný nebo přijatý AES šifrovací klíč.

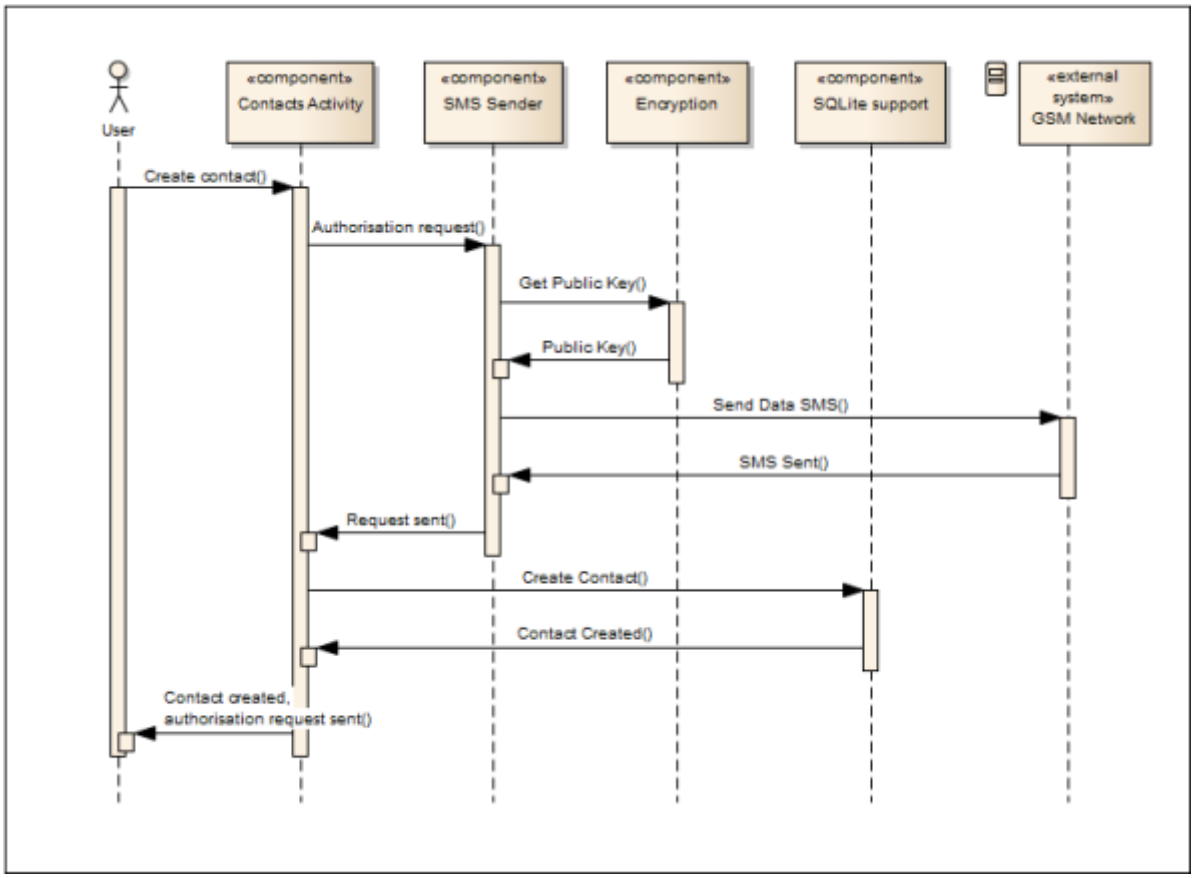
SMS Zprávy jsou ukládány do tabulky *messages*. Ve sloupci *inbox* je uložen příznak, zda se jedná o odeslanou nebo přijatou zprávu, *address* je telefonní číslo, *timestamp* obsahuje datum a čas odeslání či příjmu zprávy, *content* je samotným obsahem SMS zprávy a *unreaded* je příznak, zda byla přijatá zpráva přečtena.

5.3 Případy užití

Aplikace bude umožňovat tři základní druhy užití, které popíši v této kapitole. Dalšími jednoduchými případy užití jsou zobrazení nápovědy, mazání kontaktů, celých konverzací a zpráv.

5.3.1 Přidání nového kontaktu

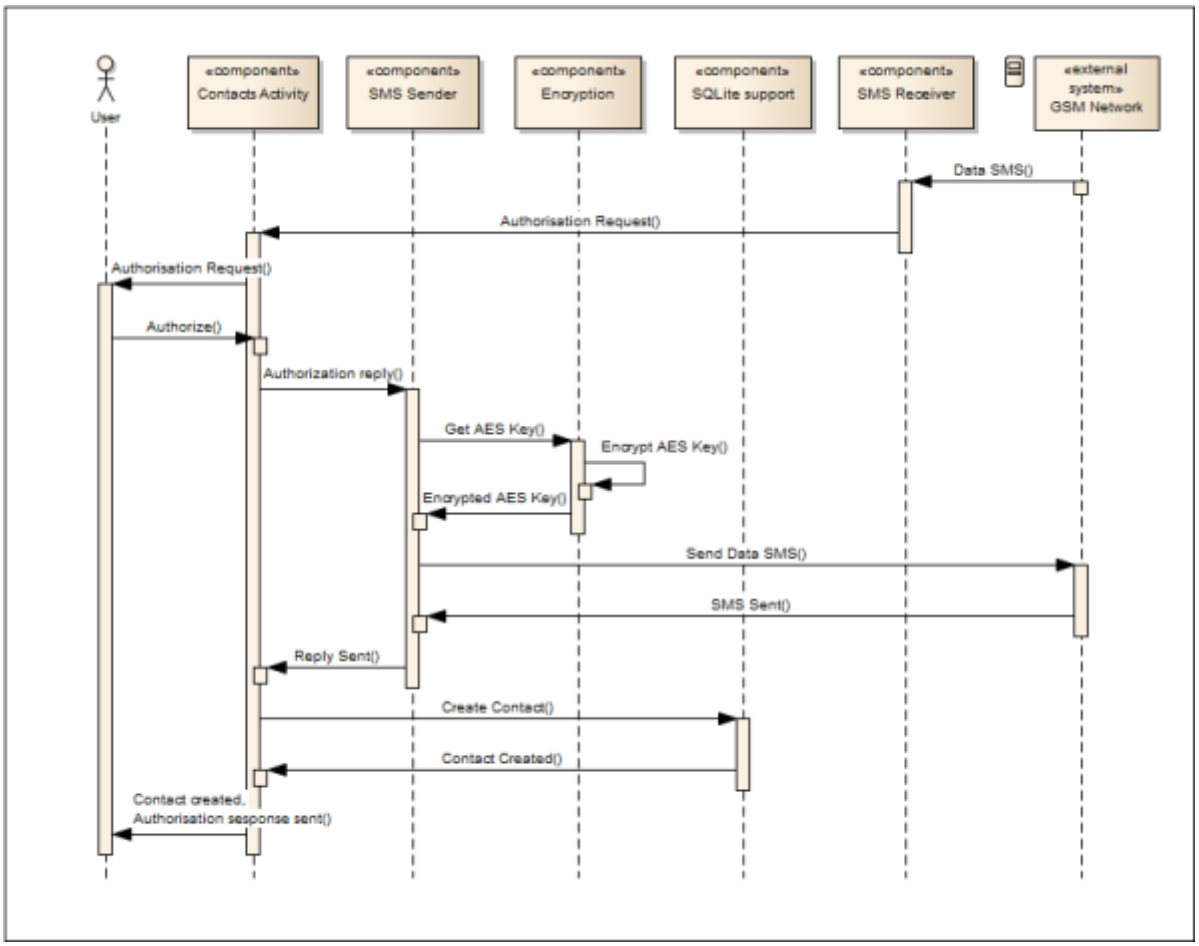
Na obrázku 5.4 je schematicky znázorněno přidání nového kontaktu. Uživatel nejprve na příslušné obrazovce uživatelského rozhraní vyplní údaje o kontaktu. Tyto informace jsou předány komponentě SMS Sender. Ta načte z databáze veřejnou část RSA klíče, převede do bytového pole, vloží společně s hlavičkou do SMS zprávy a odešle. Po úspěšném odeslání zprávy je vytvořen kontakt a uložen do databáze. Poté je uživatel informován o úspěšném vytvoření kontaktu.



Obrázek 5.4: Případ užití - přidání nového kontaktu.

5.3.2 Autorizace kontaktu

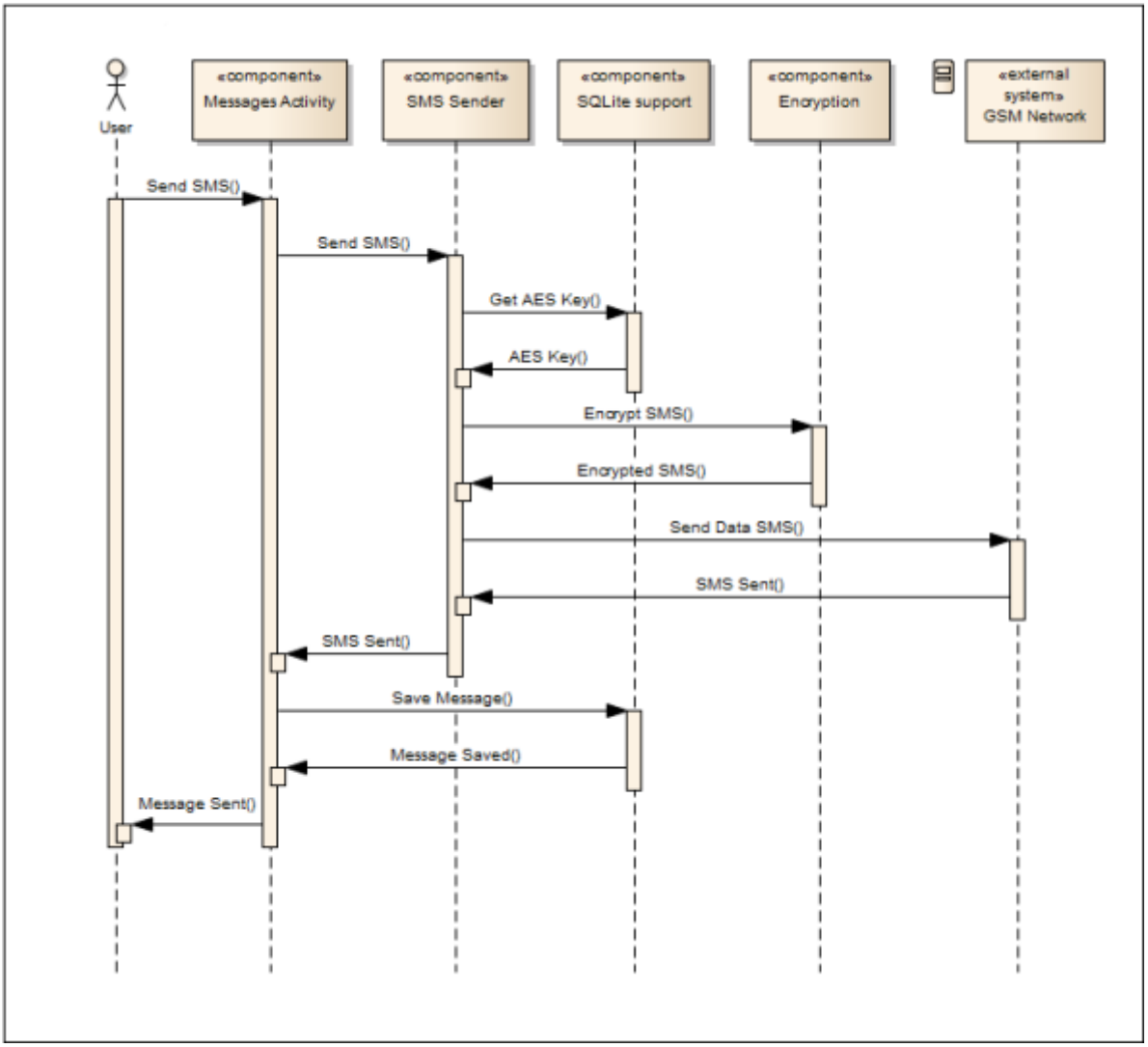
Schéma na obrázku 5.5 popisuje autorizaci kontaktu. Komponenta SMS Receiver podle hlavičky zprávy zjistí, že se jedná o požadavek na autorizaci. Uživatel je informován pomocí notifikace. Vyplní údaje o kontaktu (pouze jméno, telefonní číslo je známé z příchozí zprávy). Dále komponenta SMS Sender vyžádá od komponenty Encryption nový AES šifrovací klíč, který je vygenerován, převeden do bytového pole a zašifrován pomocí veřejné části RSA klíče z příchozí zprávy. Zašifrovaný AES klíč je spolu s hlavičkou vložen do SMS zprávy a odeslán zpět. Po úspěšném odeslání SMS zprávy je vytvořen kontakt, uložen do databáze společně s vygenerovaným AES šifrovacím klíčem a uživatel je informován o úspěšné autorizaci a vytvoření kontaktu. Pokud již kontakt existuje, je pouze aktualizován.



Obrázek 5.5: Případ užití - autorizace kontaktu.

5.3.3 Odeslání šifrované SMS zprávy

Odeslat zabezpečenou šifrovanou SMS zprávu lze pouze kontaktu, který má odesílatel v seznamu kontaktů a byl-li příjemcem autorizován. Schéma je znázorněno na obrázku 5.6.



Obrázek 5.6: Případ užití - odeslání šifrované SMS zprávy.

Uživatel vybere příjemce zprávy a napíše text zprávy. Tyto informace jsou předány komponentě SMS Sender. V databázi je nalezen AES šifrovací klíč pro daný kontakt, text zprávy je zašifrován a společně s hlavičkou je vložen do SMS zprávy a odeslán příjemci. Zpráva je poté uložena do databáze a uživatel je informován o úspěšném odeslání zprávy.

6 Implementace aplikace

Aplikace byla implementována s použitím Android Developer Tools ve vývojovém prostředí Eclipse. Z důvodu nedostupnosti open source licencí nástrojů pro multiplatformní vývoj je aplikace programována jako nativní v jazyce Java.

Jak již bylo zmíněno v kapitole 2.2, bylo nutno zvolit kompromis mezi nejnovějšími technologiemi dostupnými v posledních verzích Android API a množstvím zařízení, na kterých bude možno aplikaci používat. Jako minimální verzi API, na kterém bude aplikace spustitelná, jsem zvolil verzi 9 (Android 2.3 Gingerbread), viz tabulka 2.1. Nižší verze API jsou již velmi málo používané, nová zařízení s těmito verzemi systému již nejsou delší dobu na trhu dostupná. Nové verze API jsou vždy zpětně kompatibilní s verzemi staršími, proto není aplikace omezena žádnou nejvyšší verzí Android API.

Program se skládá ze tří hlavních vrstev, zmíněných v kapitole 5.1. Pro uživatele nejpodstatnější je prezentační vrstva, zobrazující informace pomocí grafického uživatelského rozhraní (GUI). Všechny komponenty GUI jsou závislé na zdroji dat, jímž je SQLite databáze a komponenta aplikační vrstvy SQLite Support, která data pro prezentační vrstvu zprostředkovává.

Zdrojové kódy tříd aplikace jsou umístěny v adresáři `/src/package_name`. Package_name je název balíčku aplikace a musí být unikátní v rámci celého systému Android. Pro tuto aplikaci jsem zvolil název balíčku `cz.xtvrdo00.encryptedsms`.

6.1 Datová vrstva

Přístup k databázi zprostředkovává abstraktní třída API platformy Android `android.database.sqlite.SQLiteOpenHelper`. Kromě otevření databáze pro čtení či zápis poskytuje metody pro její vytvoření v případě, že databáze neexistuje a provedení upgradu pokud je databáze aplikace ve starší verzi, než je definováno v aplikaci. Databáze není vytvořena explicitně při instalaci aplikace, ale až při vytvoření první instance třídy odvozené z `SQLiteOpenHelper`.

Z této třídy je nutno odvodit vlastní třídu a implementovat abstraktní metodu `onCreate()`, která je vyvolána v případě, že databáze neexistuje. Tato metoda vytvoří novou databázi a její strukturu, popsanou v kapitole 5.2. Zároveň při vytvoření je do tabulky *settings* vložen vygenerovaný klíčový pár pro RSA šifrovací algoritmus. Toto je časově náročná operace a proto se provádí pouze jednou při prvním přístupu k databázi. Dále je třeba implementovat abstraktní metodu `onUpgrade()`, která provede operace potřebné při přechodu na vyšší verzi. Aktuální verze je uložena v databázi v systémových tabulkách, nová verze je definována v konstruktoru třídy.

Pro samotný přístup k datům a operacím nad nimi jsem implementoval třídy `ContactsDataSource`, `MessagesDataSource` a `SettingsDataSource`. Třídy mají metody `Open()` a `Close()` pro otevření či zavření připojení k databázi a dále všechny potřebné metody pro práci s daty. Na tyto třídy se budu odkazovat dále v popisu implementace.

6.2 Prezentační vrstva

Data a funkcionalita aplikace jsou uživateli přístupné prostřednictvím grafického uživatelského rozhraní. To se na platformě Android skládá z jednotlivých obrazovek nazývaných *Activity*. Základní princip je popsán v kapitole 2.3.

Jednotlivé aktivity jsou podtřídami abstraktní třídy `android.app.Activity`. Jejich grafické rozvržení je definováno v příslušném značkovacím XML souboru v adresáři `/res/layout/`. Velikosti a rozložení komponent nejsou definovány absolutně ale relativně k dostupné ploše na obrazovce. Tím je docíleno správné zobrazení obrazovky na zařízeních s různými velikostmi a rozlišeními obrazovky. Tato aplikace obsahuje čtyři základní aktivity. Seznam konverzací, seznam zpráv, seznam kontaktů a editační okno kontaktu. Následující fragment kódu ukazuje popis rozvržení komponent pro aktivitu seznamu konverzací, ve zdrojových kódech se jedná o soubor `activity_conversations.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Button
        android:id="@+id/btn_newMessage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/button_new_message"
        android:onClick="newMessage_onClick"/>

    <ListView
        android:id="@+id/list_conversations"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Základním kořenovým elementem je vždy typ rozložení komponent na daném okně. Jednotlivé typy formátování mohou být stromovitě zanořeny pro dosažení požadovaného vzhledu a rozmístění komponent.

Jednotlivé grafické prvky (označovány *View*) mají přiřazený unikátní číselný identifikátor `id`, který je generován při překladu aplikace. Pro přístup k jednotlivým komponentám se konkrétní instance vyhledá pomocí textového popisu atributu `id`.

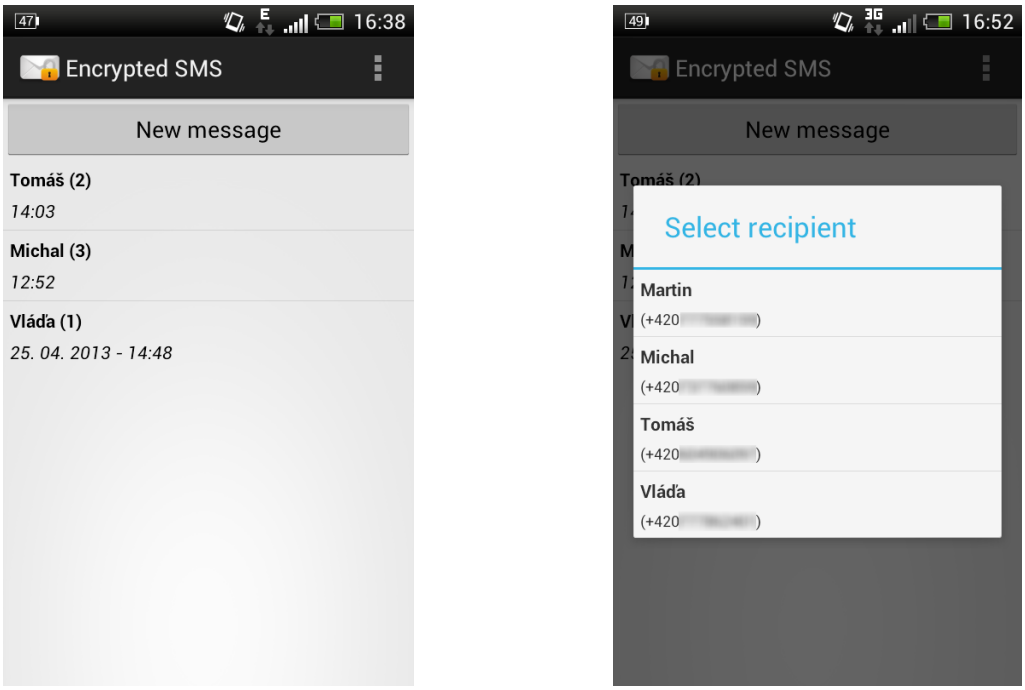
Veškeré textové popisy a jiné textové informace jsou definovány v XML souboru `/res/values/strings.xml`. Je to z toho důvodu, že texty mohou být přeložené do více jazyků. Pro další jazyk se vytvoří nová složka, například `/res/values-cz/` do které se umístí lokalizovaný seznam řetězců `strings.xml`. Aplikace využije řetězce ze složky příslušné lokalizaci nastavené v mobilním zařízení. Pokud taková lokalizace není k dispozici, využijí se hodnoty v implicitní složce `/res/values/`. Dále je možno nastavit konkrétní lokalizaci přímo v aplikaci, například pomocí nastavení. Na výše uvedeném příkladu je patrné, jakým způsobem se přiřazují lokalizovatelné texty, zde například pro text tlačítka. Je samozřejmě možné definovat text přímo, bez použití tabulky řetězců, ale programátor se tak zbavuje možnosti daný text lokalizovat.

Interakce s prvky uživatelského rozhraní lze nastavit v kódu aktivity, nebo také ve značkovacím popisu v XML formátu. Na výše uvedeném příkladu je akci tlačítka `onClick` (stisknutí tlačítka) přiřazena obslužná metoda `newMessage_onClick`, která musí být v kódu dané aktivity implementována.

Každá aktivita může implementovat několik virtuálních metod abstraktní třídy `Activity`. Metoda `onCreate()` je vyvolána při vytvoření nové instance dané aktivity. Inicializují se zde prvky grafického uživatelského rozhraní a přiřazují obslužné metody pro různé akce. Metoda `onPause()` je zavolána systémem při otevření jiné aktivity. Ruší se zde například registrace na různé události. Důležitou metodou je také `onResume()`, která je vyvolána při každém přenesení dané aktivity do popředí. Používá se k re-inicializaci dat pro aktivitu a registraci na různé události.

6.2.1 Seznam konverzací

Seznam konverzací, viz obrázek 6.1(a), je hlavní aktivitou aplikace. Zobrazují se zde konverzace se všemi kontakty. Pro seznam je použita komponenta `ListView`. Zobrazení seznamu dat v této komponentě zprostředkovává generická třída `android.widget.ArrayAdapter<T>`. Lze použít přímo instance této třídy, která v jednotlivých polích seznamu zobrazí text získaný pomocí metody `toString()` zavolané nad všemi prvky seznamu objektů generického typu.



(a) Seznam konverzací (b) Výběr příjemce nové zprávy

Obrázek 6.1: Aktivita seznam konverzací.

Pro zobrazení informací o konverzacích v podobě zobrazené na obrázku 6.1(a) jsem odvodil vlastní třídu `ConversationsArrayAdapter` z třídy `ArrayAdapter<Conversation>`. Tato třída implementuje abstraktní metodu getView(), která z poskytnuté instance třídy `Conversation` vytvoří příslušný řádek pro seznam konverzací. Rozložení řádku je definováno v souboru `/res/layout/conversation_list_item.xml`. Pro každou konverzaci je zde zobrazen název kontaktu (případně telefonní číslo, pokud byl kontakt smazán), počet zpráv v konverzaci (součet počtu všech odchozích a příchozích zpráv) a čas poslední zprávy. V případě, že poslední zpráva je starší než z aktuálního dne, je zobrazen i datum. Datum a čas jsou zobrazovány ve formátu definovaném v nastavení mobilního zařízení.

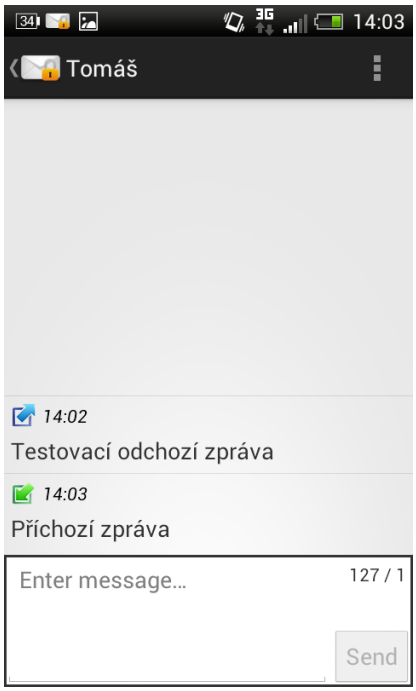
Dlouhým stiskem nad některou z konverzací je vyvoláno kontextové menu s možnostmi pro otevření či smazání konverzace.

Data pro seznam konverzací jsou získávána pomocí metody třídy `MessagesDataSource` z databáze. Tato aktivita je registrovaná na událost `MessagesUpdated`, která je vyvolána při každé změně v databázové tabulce *messages*. Při obdržení této události je znovu načten seznam a konverzace obsahující nepřečtené zprávy jsou zvýrazněny jinou barvou textu pro lepší orientaci uživatele.

Tlačítko pro vytvoření nové zprávy je umístěno nad seznamem konverzací, po jeho stisknutí zobrazí dialog se seznamem autorizovaných kontaktů, jimž lze odeslat šifrovanou SMS zprávu, viz obrázek 6.1(b). Po vybrání jednoho z nich je zobrazena aktivita seznamu zpráv s vybraným kontaktem.

6.2.2 Seznam zpráv

Ze seznamu konverzací lze kliknutím na jednu z nich přejít na seznam zpráv s daným kontaktem, viz obrázek 6.2. Zprávy jsou opět načítány pomocí metod třídy `MessagesDataSource` a zobrazovány v komponentě `ListView` pomocí adaptéru `MessagesArrayAdapter`. Ikona označuje, zda se jedná o zprávu přijatou či odeslanou; čas, případně i datum jsou zobrazeny stejným způsobem jako v seznamu konverzací. Na dalším řádku následuje samotný text zprávy. Kontextové menu pro každou ze zobrazených zpráv umožňuje tuto zprávu vymazat.

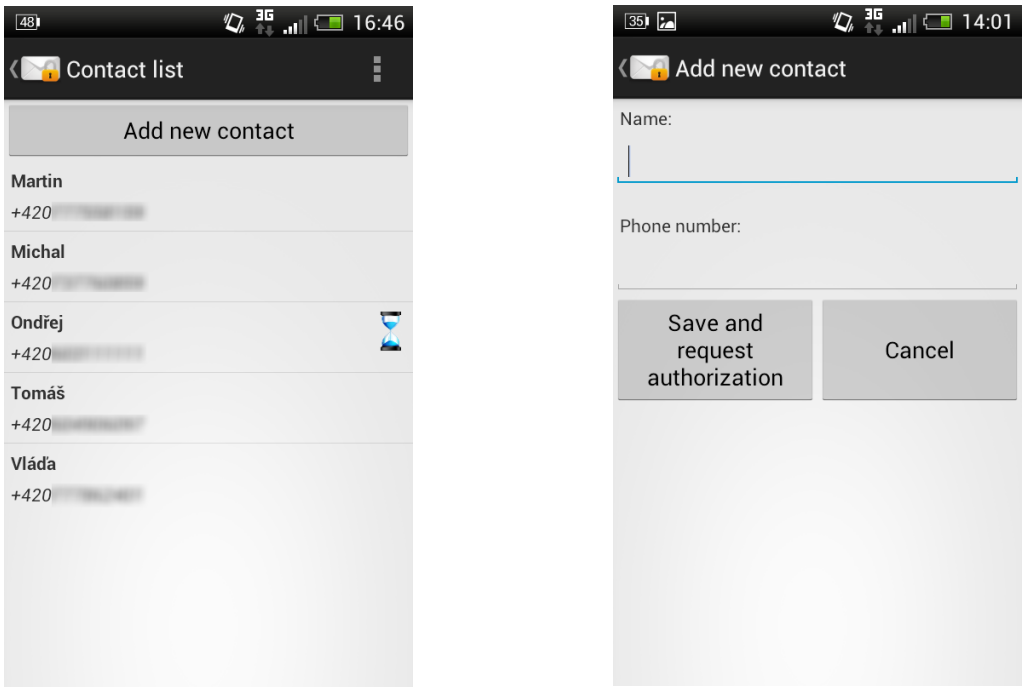


Obrázek 6.2: Aktivita seznam zpráv.

Pod seznamem zpráv je textové pole (komponenta *EditText*), do kterého lze napsat zprávu určenou k odeslání danému kontaktu. Toto textové pole je neaktivní v případě, že daný kontakt není autorizován. Po napsání zprávy ji lze odeslat příslušným tlačítkem. Nad tlačítkem je zobrazeno počítadlo napsaných znaků. Zpráva může být dlouhá maximálně 127 bytů (vysvětleno v kapitole 6.3.2). Psaný text je převáděn do pole bytů pomocí kódování UTF-8. Znak bez diakritiky je reprezentován jedním bytem, znak s diakritikou dvěma byty. Zbývající počet znaků je zobrazován uživateli, za lomítkem je potom celkový počet zpráv, které budou odeslány. Je tak možno posílat zprávy delší než 127 znaků (bytů).

6.2.3 Seznam kontaktů

Seznam kontaktů, viz obrázek 6.3(a), je přístupný z menu aplikace. Zobrazuje informace o přidaných kontaktech. Pro databázové operace s kontakty jsou použity metody třídy `ContactsDataSource`, pro zobrazení položek seznamu adaptér `ContactsArrayAdapter`. U přidaných kontaktů, kterými nebyla dosud potvrzena autorizace, je zobrazena ikona přesýpacích hodin. Těmto kontaktům nelze zasílat SMS zprávy. Kliknutím na vybraný kontakt se spustí aktivita seznamu zpráv pro daný kontakt. Dlouhým stiskem nad některým z kontaktů je zobrazeno kontextové menu nabízející editaci kontaktu případně jeho smazání.



(a) Seznam kontaktů (b) Editace kontaktu

Obrázek 6.3: Seznam kontaktů.

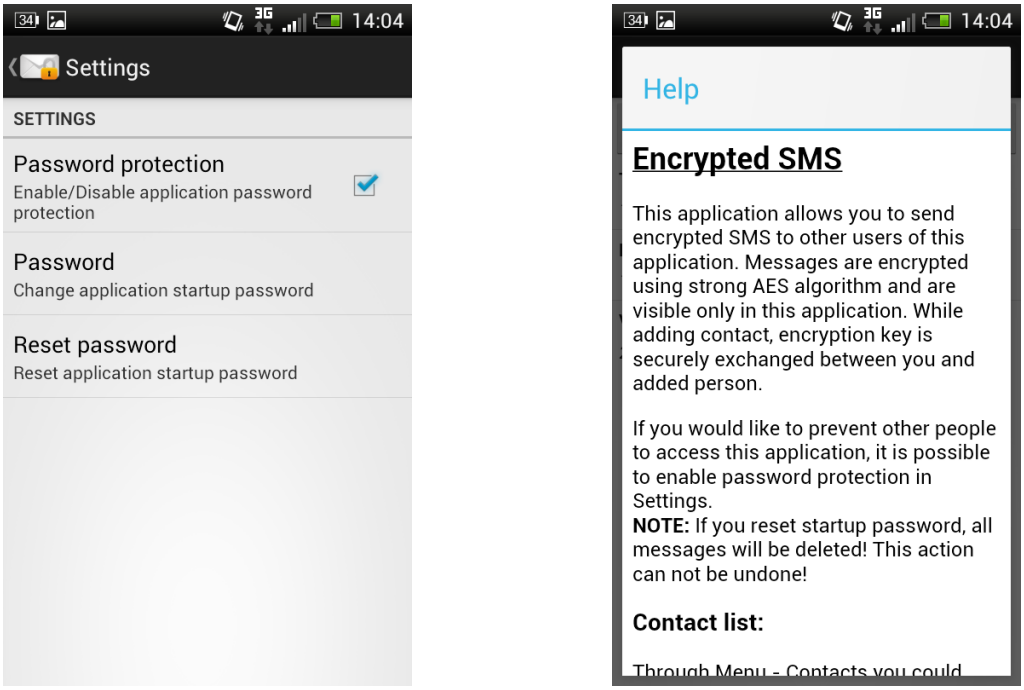
Přidání nového kontaktu, editace kontaktu a potvrzení autorizace zprostředkovává uživateli aktivita `ContactEditActivity`. Vzhled aktivity je patrný z obrázku 6.3(b). Dle dané funkcionality má potvrzovací tlačítko příslušný text.

6.2.4 Nastavení systému, nápověda

Z menu aplikace je kromě seznamu kontaktů přístupné také nastavení, viz obrázek 6.4(a). Je možno zde zapnout ochranu aplikace heslem. Při spuštění aplikace, případně při jejím obnovení po delší době nečinnosti (5 minut) bude po uživateli vyžadováno heslo.

Tato funkcionalita je implementována v hlavní třídě aplikace, `MyApplication` odvozené od třídy `android.app.Application`. Jsou zde metody pro aktualizaci času poslední interakce s aplikací a dialog pro zadání hesla. Tyto metody jsou využívány všemi aktivitami v metodách `onPause()` a `onResume()`. Heslo lze změnit a také resetovat (například při zapomenutí uživatelem).

Při resetu hesla dojde se smazání všech SMS zpráv v aplikaci, uživatel je příslušným dialogem varován.



(a) Aktivita nastavení aplikace (b) Nápověda

Obrázek 6.4: Nastavení aplikace.

Další položkou přístupnou z menu aplikace je nápověda, viz obrázek 6.4(b), která je zobrazena v dialogu. Jedná se o formátovaný text ve značkovacím jazyce HTML, uložený v souboru `/res/raw/help.html`. Nápovědu lze také lokalizovat.

6.3 Aplikační vrstva

Jednu z komponent aplikační vrstvy, SQLite Support jsem popsal společně s databází v kapitole 6.1. Dalšími částmi jsou odesílání a příjem SMS zpráv a podpora pro šifrování.

6.3.1 Odesílání a příjem SMS zpráv

Metody pro odesílání zpráv jsem implementoval v třídě `SMSSender`. Zprávy jsou odesílány s pomocí systémové třídy `android.telephony.SmsManager` metodou `sendDataMessage`. Tato metoda umožňuje odesílat SMS zprávy obsahující libovolná data až do velikosti 133 bytů. Tyto zprávy jsou odeslány s příznakem cílového portu. Jedná se o virtuální port, na kterém musí být na cílovém zařízení zaregistrován poslouchající proces. Pokud není registrován žádný přijímající proces, přijatá zpráva není zpracována a je systémem ignorována. Všechny metody mají jako jeden z parametrů odkaz na metodu, která je vyvolána po pokusu o odeslání zprávy, s parametrem případného chybového kódu. Tohoto využívá prezentační vrstva pro informování uživatele, zda byla zpráva odeslána či nikoliv.

Pro rozpoznání typu odesílaná zprávy vkládají metody před samotná data hlavičku s nejnutnějšími informacemi, viz tabulka 6.1.

Tabulka 6.1: Hlavičky datových zpráv.

Typ zprávy	Hlavička
Požadavek na autorizaci	Byte[1] – (0)
Potvrzení autorizace	Byte[1] – (1)
Šifrovaná SMS zpráva	Byte[3] – (2, A, B)

První byte v hlavičce je vždy označení typu zprávy. Pro šifrovanou SMS zprávu následují dva byty označující celkový počet zpráv a pořadí dané zprávy. Tato informace je zde pro účely zasílání zpráv delších než 127 znaků, které jsou rozděleny do více fyzických zpráv. Druhý byte (A) značí pořadí zprávy, třetí byte (B) potom celkový počet zpráv.

Požadavek na autorizaci obsahuje za hlavičkou veřejný RSA klíč odesílatele převedený do pole bytů. Potvrzení autorizace obsahuje AES šifrovací klíč zašifrovaný obdrženým veřejným RSA klíčem. Samotná šifrovaná SMS zpráva obsahuje uživatelem zadaný text, převedený do pole bytů v kódování UTF-8 zašifrovaný algoritmem AES.

Pro příjem zpráv slouží třída `IncomingSMSReceiver` odvozená z jedné ze základních komponent `android.content.BroadcastReceiver`. Tato třída je zaregistrována v souboru `AndroidManifest.xml` pro příchozí datové SMS zprávy na specifikovaném čísle portu. Obsluha události musím proběhnout v krátkém čase, proto tato třída pouze načte pole bytů z příchozí zprávy a spustí službu, implementovanou v třídě `SMSReceiverService`, která zajistí zpracování příchozích dat. Podle údaje v hlavičce zprávy je zjištěn typ zprávy, data jsou uložena do databáze a uživatel je informován pomocí systémového oznámení o příjmu zprávy.

6.3.2 Podpora pro šifrování

Pro šifrování jsou použity algoritmy RSA a AES. Metody pro generování klíčů, jejich převod do bytové podoby a samotné šifrování jsou implementovány v třídě `EncryptionSupport`. Tato třída využívá metody z knihovny jazyka Java `javax.crypto`. Algoritmus RSA slouží pro bezpečnou výměnu klíčů pro symetrickou šifru AES, pomocí které jsou poté šifrovány uživatelské zprávy.

RSA algoritmus používá klíč o délce 1024 bitů. To z důvodu, aby výsledný zašifrovaný blok nepřesál 133 bytů a vešel se do jedné SMS zprávy. Délka bloku šifrované zprávy odpovídá délce klíče. Zašifrovaný AES šifrovací klíč má délku 128 bytů a i s jedním přidaným bytem pro hlavičku je kratší než zmiňovaných 133 bytů.

Uživatelem odesílané SMS zprávy jsou šifrovány symetrickým algoritmem AES. Jak již bylo uvedeno v kapitole 3.1.3, tento algoritmus pracuje s bloky dat o velikosti 16 bytů. Na konec otevřeného textu je vždy přidáván jeden zarovnávací byte. Maximální počet bloků, které se vejdou do jené SMS zprávy je 8, což odpovídá 128 bytům zprávy. Po odečtení povinného zarovnávacího bytu na konci zprávy je tak možné použít 127 bytů otevřeného textu. Z tohoto důvodu je délka uživatelské zprávy omezena na 127 bytů. Před zašifrovanou zprávu o délce maximálně 128 bytů jsou přidány tři byty hlavičky a zpráva je odeslána pomocí jedné datové SMS.

7 Testování aplikace

Testování aplikace v době vývoje probíhalo pomocí emulátoru mobilního telefonu se systémem Android, dostupného z Android SDK. Je možno vytvořit libovolné množství emulátorů s různými verzemi Android API. Tento postup je vhodný pro ladění a testování grafického uživatelského rozhraní díky možnosti testovat veškeré změny na více různých verzích platformy Android.

Testování samotné funkcionality aplikace nebylo na emulátoru možné. Současná nejvyšší verze emulátoru neumožňuje přijímat datové SMS zprávy (debugger hlásí chyby kvůli nedostatečným oprávněním aplikace). Proto bylo nutné funkcionalitu testovat na reálných zařízeních přes GSM síť. Testování probíhalo na zařízeních HTC Desire X[7] se systémem Android 4.0.4 a LG P990 Optimus 2X[8] se systémem Android 2.3.3. Bylo tak možno ověřit kompatibilitu datových SMS zpráv mezi zařízeními různých výrobců s různými verzemi platformy Android.

[7] Informace o produktu dostupné z URL: http://www.htc.com/cz/smartphones/htc-desire-x/

[8] Informace o produktu dostupné z URL: http://www.lg.com/uk/mobile-phones/lg-P990-optimus-2x

8 Závěr

V první části jsem nejprve popsal možnosti vývoje aplikací pro chytré mobilní telefony a problematiku šifrování. Poté jsem prozkoumal aplikace, které jsou pro daný účel k dispozici. Nakonec jsem navrhl a implementoval aplikaci pro komunikaci prostřednictvím šifrovaných SMS zpráv.

K šifrování je vždy třeba použít šifrovací klíč. Možností pro výměnu klíče je několik. Uživatelé, chtějící zabezpečeně komunikovat si musí šifrovací klíč (heslo) sdělit a do aplikace zadat. Pro zachování tajemství pouze mezi těmito dvěma stranami je nutný osobní kontakt a bezchybné zadání klíčů do aplikace. Další možností je výměna klíčů mezi zařízeními například pomocí technologie Bluetooth. Tato varianta znovu vyžaduje osobní kontakt komunikujících stran.

Pro účely mobilních zařízení a s použitím dostupných technologií jsou výše zmíněné postupy nevhodné a neposkytují uživatelsky přijatelnou a zároveň bezpečnou výměnu šifrovacích klíčů.

Navrhl jsem proto systém výměny klíčů, který žádná zdarma dostupná aplikace pro chytré mobilní telefony nenabízí. Šifrovací klíč pro symetrickou šifru, která je použita pro šifrování uživatelských zpráv, je mezi dvěma zařízeními vyměněn s použitím asymetrické šifry (dvě SMS zprávy, které si aplikace vymění před zahájením uživatelské komunikace). Implementovaná aplikace toto řeší uživatelsky přívětivým a nenáročným způsobem.

Použitý systém výměny klíče a samotného šifrování uživatelské komunikace je velmi bezpečný. Jednou z možných hrozeb by bylo zachycení vyměňovaného klíče „osobou uprostřed“ (v literatuře označovanou jako „man in the middle“) odposlouchávající SMS komunikaci mezi dvěma stranami. Tato hrozba je minimalizována použitím asymetrické šifry pro výměnu klíče.

Další možností prolomení je útok hrubou silou. Vzhledem k použité symetrické šifře (algoritmus AES) s velmi dlouhým šifrovacím klíčem, který je náhodně generovaný a krátké délce zpráv je tento typ útoku s dnešními výpočetními možnostmi téměř nemožný.

Poslední hrozbou je ztráta či krádež mobilního zařízení, kde již jsou uživateli zprávy k dispozici v dešifrované podobě. Tato hrozba je minimalizována možností použít heslo pro přístup k aplikaci. Při pokusu o jeho resetování jsou všechny zprávy vymazány.

Přínosem této práce je zmapování funkcionality a praktické využití zasílání datových SMS zpráv mezi chytrými mobilními telefony, které je aplikacemi velmi málo využíváno. Navržená aplikace pak umožňuje bezpečnou výměnu klíčů, kterou žádná dostupná aplikace uživatelsky přívětivým způsobem neumožňuje.

Po dokončení byla aplikace nainstalována a několik týdnů provozována na čtyřech mobilních telefonech s různými verzemi platformy Android (2.3.3 – 4.2). Aplikaci lze pro účely bezpečné komunikace velmi dobře použít, pro uživatele je přehledná a poskytuje všechny základní možnosti nastíněné v návrhu aplikace. Po analýze zpětné vazby od uživatelů byla doimplementována podpora ochrany aplikace heslem a provedeny drobné změny v grafickém uživatelském rozhraní.

Pro další vývoj projektu bude aplikace zdarma umístěna na aplikační obchod Google play a vylepšena podle zpětné vazby uživatelů. Dále je možné aplikaci implementovat i pro jiné mobilní platformy (zejména druhou nejpoužívanější Apple iOS). Dalším možným zlepšením pro uživatele by byla možnost exportovat a importovat databázi s kontakty a zprávami a možnost přidávat kontakty ze stávajícího telefonního seznamu mobilního telefonu. Pro větší objemy dat a možnost přenosu i multimediálního obsahu je možno doimplementovat podporu dnes již často využívaných zpráv MMS.

Literatura

[1] GOOGLE INC. *Android* [online]. 2013 [cit. 2013-04-23]. Dostupné z: http://www.android.com/

[2] CIMALA, Petr. Víkendové grafy: Kdo se bojí?. *Klub investorů: Cesta k úspěchu* [online]. 2013, 8.4.2013 [cit. 2013-04-27]. Dostupné z: http://www.klubinvestoru.com/magazin/i1561-vikendove-grafy-kdo-se-boji

[3] IOS Developer Program. APPLE INC. *Developer* [online]. 2013 [cit. 2013-04-27]. Dostupné z: https://developer.apple.com/programs/ios/develop.html

[4] Android architecture. TUTORIALSPOINT. *Tutorialspoint: Simple Easy Learning* [online]. 2012 [cit. 2013-04-28]. Dostupné z: http://www.tutorialspoint.com/android/android_architecture.htm

[5] Get the Android SDK. GOOGLE INC. *Developers* [online]. 2013 [cit. 2013-05-11]. Dostupné z: http://developer.android.com/sdk/index.html

[6] GOOGLE INC. *Google play* [online]. 2013 [cit. 2013-04-23]. Dostupné z: https://play.google.com/store

[7] Application Fundamentals. GOOGLE INC. *Developers* [online]. 2013 [cit. 2013-04-28]. Dostupné z: https://developer.android.com/guide/components/fundamentals.html

[8] Dashboard. GOOGLE INC. *Developers* [online]. 2013, 2.4.2013 [cit. 2013-04-28]. Dostupné z: http://developer.android.com/about/dashboards/index.html

[9] VÁVRŮ, Jiří. *IPhone: vývoj aplikací*. 1. vyd. Praha: Grada, 2012, 179 s. Průvodce (Grada). ISBN 978-80-247-4457-5.

[10] XAMARIN INC. *Xamarin: Create iOS, Android, Mac and Windows apps in C#* [online]. 2013 [cit. 2013-04-29]. Dostupné z: http://xamarin.com/

[11] DOSEDĚL, Tomáš. *Počítačová bezpečnost a ochrana dat*. Vyd. 1. Brno: Computer Press, 2004, 190 s. ISBN 80-251-0106-1.

[12] PIPER, F a Sean MURPHY. *Kryptografie*. 1. vyd. v českém jazyce. Překlad Pavel Mondschein. Praha: Dokořán, 2006, 157 s. ISBN 80-736-3074-5.

[13] MIČKA, Pavel. Algoritmus RSA. *Algoritmy.net: příručka vývojáře* [online]. 2011 [cit. 2013-04-30]. Dostupné z: http://www.algoritmy.net/article/4033/RSA

[14] Google play: Secure SMS. GOOGLE INC. *Google play* [online]. 2012, 16.7.2012 [cit. 2013-04-23]. Dostupné z: https://play.google.com/store/apps/details?id=com.secsms

[15] Google play: Secure SMS. GOOGLE INC. *Google play* [online]. 2012, 6.3.2012 [cit. 2013-04-23]. Dostupné z: https://play.google.com/store/apps/details?id=legresoft.encryption

[16] *SQLite* [online]. 2013 [cit. 2013-05-01]. Dostupné z: http://www.sqlite.org/

Seznam příloh

Příloha 1 – Obsah CD

- Zdrojové texty programu
- Instalační soubor aplikace (formát .apk)
- Manuál (formát .pdf)
- Text této práce (formáty .doc a .pdf)

Příloha 2 – Manuál

Instalace aplikace

Aplikaci lze nainstalovat na zařízení s operačním systémem Android ve verzi 2.3 (Gingerbread) a novější.

Pro instalaci aplikací z jiného zdroje než je obchod Google Play je nutno v zařízení povolit instalaci z externích zdrojů. Tuto volbu naleznete v menu zařízení (pro Android verze 4.x v menu *Nastavení → Zabezpečení → Neznámé zdroje*).

Po povolení instalace externích aplikací nakopírujte instalační soubor na paměťovou kartu zařízení a nainstalujte aplikaci pomocí libovolného správce souborů. Před instalací je nutné potvrdit práva, která aplikace k běhu vyžaduje.

Použití aplikace

Aplikace umožňuje šifrovanou komunikaci pomocí SMS zpráv. Všechna zařízení, mezi nimiž chcete zasílat zabezpečené zprávy, musí mít tuto aplikaci nainstalovánu! Při přidání kontaktu dojde k automatické výměně šifrovacích klíčů.

V menu aplikace jsou k dispozici tyto volby:

- *Contacts* – Seznam kontaktů
- *Settings* – Nastavení aplikace
- *Help* – Stručná nápověda

Přidání kontaktu

Administrace kontaktů je k dispozici v menu *Contacts*. Pro přidání kontaktu vyberte *Add new contact*. Po vyplnění jména (*Name*) a telefonního čísla (*Phone number*) vyberte volbu *Save and request authorisation*. Tímto krokem odešlete žádost o autorizaci přidávané osobě pomocí jedné SMS zprávy. Nový kontakt je zobrazen v seznamu kontaktů a až do potvrzení autorizace je u něho zobrazena ikona přesýpacích hodin a nelze tomuto kontaktu odesílat SMS zprávy.

Volby Odebrání kontaktu a jeho editace jsou dostupné z kontextového menu (dlouhý stisk nad vybraným kontaktem). Editovat lze pouze jméno, pro změnu telefonního čísla je nutné kontakt znovu vytvořit a odeslat novou žádost o autorizaci.

Autorizace kontaktu

Přijetí žádosti o autorizaci je oznámeno klasickým systémovým oznámením (podobné například oznámení o příchozí zprávě či zmeškaném hovoru). Po vybrání oznámení vyplňte jméno kontaktu (pokud se jedná o žádost od neznámého kontaktu), telefonní číslo je vyplněno automaticky. Volbou *Authorize* odešlete potvrzení autorizace pomocí jedné SMS zprávy.

Přijaté potvrzení autorizace se zobrazí v systémových oznámeních. U kontaktu zmizí ikona přesýpacích hodin a je umožněno posílat SMS zprávy.

Odesílání zabezpečených SMS zpráv

Pro odesílání zpráv musí být příjemce v seznamu kontaktů a musí potvrdit autorizaci.

Vybráním libovolné konverzace ze seznamu konverzací se dostanete k seznamu zpráv. Zde vidíte všechny dosud odeslané a přijaté SMS zprávy. Do seznamu zpráv se můžete dostat i stiskem tlačítka *New message* ze seznamu konverzací nebo vybráním kontaktu ze seznamu kontaktů.

Na této obrazovce, v její dolní části, je možno zadat text odesílané zprávy a tuto odeslat tlačítkem *Send*. Nad tlačítkem odeslání zprávy je zobrazen počet zbývajících znaků v jedné SMS a celkový počet odesílaných zpráv. Při psaní textu se tato počítadla dynamicky aktualizují.

Přijatá SMS zpráva je oznámena příjemci systémovým oznámením, jehož vybráním se aplikace otevře na seznamu zpráv s daným kontaktem.

Nastavení aplikace

Z menu je přístupná položka nastavení (Settings). Zde je možno zapnout ochranu aplikace heslem, které je poté vyžadováno při spuštění aplikace či po delší době nečinnosti. Toto heslo je možno změnit, případně deaktivovat.